ProbGraph: High-Performance and High-Accuracy Graph Mining with Probabilistic Set Representations

Maciej Besta[1†], Cesare Miglioli[2], Paolo Sylos Labini[3], Jakub Tětek[4], Patrick Iff[1],
Raghavendra Kanakagiri[5], Saleh Ashkboos[1], Kacper Janda[6], Michał Podstawski[7,8],
Grzegorz Kwaśniewski[1], Niels Gleinig[1], Flavio Vella[9], Onur Mutlu[1], Torsten Hoefler[1†]

[1]ETH Zurich [2]Research Center for Statistics, University of Geneva [3]Free University of Bozen-Bolzano
[4]BARC, University of Copenhagen [5]UIUC [6]AGH-UST [7]Warsaw University of Technology
[8]TCL Research Europe [9]University of Trento †Corresponding authors

Abstract—**Important graph mining problems such as Clustering are computationally demanding. To significantly accelerate these problems, we propose ProbGraph: a graph representation that enables simple and fast approximate parallel graph mining with strong theoretical guarantees on work, depth, *and* result accuracy. The key idea is to represent sets of vertices using probabilistic set representations such as Bloom filters. These representations are much faster to process than the original vertex sets thanks to vectorizability and small size. We use these representations as building blocks in important parallel graph mining algorithms such as Clique Counting or Clustering. When enhanced with ProbGraph, these algorithms significantly outperform tuned parallel exact baselines (up to nearly 50× on 32 cores) while ensuring accuracy of more than 90% for many input graph datasets. Our novel bounds and algorithms based on probabilistic set representations with desirable statistical properties are of separate interest for the data analytics community.**

Index Terms—**Approximate Graph Mining, Approximate Graph Pattern Matching, Approximate Triangle Counting, Approximate Community Detection, Approximate Graph Clustering, Bloom Filters, MinHash, K Minimum Values, High-Performance Graph Computations, Graph Sketching**

I. Introduction

Graph mining is an important part of the graph processing landscape, with problems related to discovering patterns in graphs, for example Clustering, Clique Counting, or Link Prediction [1]–[4]. Accelerating graph mining is notoriously difficult because these problems are hard to parallelize due to properties such as high irregularity or little locality [5]–[10]. Simultaneously, graph mining underlies many important computational problems in social network analysis, machine learning, computational sciences, and others [11]–[14].

In approximate computing, a certain (ideally small) amount of accuracy is sacrificed, in exchange for speedups or reduced energy consumption [15]–[17]. It relaxes the need for full precision at the level of arithmetic blocks, processing units, pertinent error and quality measures, algorithms, programming models, and many others. Traditionally, graph algorithms with provable approximation ratios were developed to alleviate the hardness of various NP-Complete graph problems, such as minimum graph coloring [18], [19]. Still, these works are usually complex, specific to a particular graph problem, and often need additional heuristics to be easily used in practice.

Moreover, there are many heuristics for approximating graph properties such as betweenness centrality [20]–[25], minimum spanning tree weight [26], maximum matching [27], reachability [28], graph diameter [29], [30], and others [30]–[34]. *Unfortunately, these schemes are all specific to a particular graph problem or algorithm.*

To alleviate the above-mentioned issues with accelerating graph mining, we propose ProbGraph (PG), a probabilistic graph representation for simple, versatile, fast, and tunable approximate graph mining. We observe that the *input graph and many auxiliary data structures are effectively a collection of sets of vertices and edges* [1], [4]. Here, our key idea is to encode such sets with carefully selected **probabilistic set representations** (sometimes called set sketches) such as Bloom filters [35]. This results in a "probabilistic" graph representation that – as we will show – can accelerate different graph algorithms at a (tunable) accuracy-storage-performance tradeoff. Importantly, sets and set operations are common in graph problems, making PG applicable to many algorithms.

We first show that many time-consuming operations in different graph algorithms can be expressed with *set intersection cardinality* $|X \cap Y|$. For example, deriving common parts of vertex neighborhoods takes more than 90% of the time in common Triangle Counting algorithms [1], [4], [36], and it can be expressed as a sum of $|X \cap Y|$ (over different X and Y). *We identify more such graph algorithms.*

Second, we carefully select three probabilistic set representations: Bloom filters (BF) [35] and two types of MinHash (MH) [37]. We use these set representations to design *estimators* $\widehat{|X \cap Y|}$ that approximate $|X \cap Y|$. Our central motivation is that these estimators are much faster to obtain than the exact $|X \cap Y|$ thanks to performance-friendly properties. We conduct a *work-depth* analysis, *showing formally that ProbGraph-enhanced graph algorithms have abundant parallelism*. For example, X and Y, when represented with Bloom filters, are bit vectors. Thus, $|X \cap Y|$ amounts to computing a bitwise AND, followed by a reduction. Such an operation *significantly benefits from vectorization*. Moreover, thanks to its fixed-size set representations, ProbGraph *exhibits excellent load balancing properties*: all set intersections are conducted over the same size bit vectors, annihilating issues related to intersecting neighborhoods of different sizes. This is particularly attractive as modern graph datasets have very often high skews in degree distributions [7].

Importantly, *we ensure that our estimators have strong theoretical guarantees on their accuracy (i.e., quality)*. For this, we develop or adapt bounds on the quality of estimators $\widehat{|X \cap Y|}$. Here, we prove how far they deviate from the true size of the set intersection, and we provide upper bounds for their mean squared error (MSE). We also produce quality bounds that are better than past works. To derive these quality bounds, we use concentration inequalities and other statistical concepts such as sub-Gaussian random variables [38]. For example, the probability that our estimator based on MH deviates from the true value by more than a given distance t, decreases *exponentially* with t.

Moreover, we show that, for some representations, we offer *Maximum Likelihood Estimators* (MLE) [39]. Thus, these estimators are asymptotically[1] unbiased (the expected estimator value converges to the true parameter value) and efficient (asymptotically, *no other estimator* has lower variance). Due to the prevalence of BF and MH in high-performance data mining, our novel results on the theory of probabilistic set representations are of interest beyond graph analytics.

Our fast parallel implementation of PG enables important graph mining problems (TC, clustering [40]–[42], 4–clique counting [43], and vertex similarity [44], [45]) to achieve very large performance advantages over tuned baselines, even up to 50× lower runtimes when using 32 cores. This is caused by the fact that both work and depth of ProbGraph-enhanced graph algorithms are lower than those of the exact tuned baselines. Simultaneously, using small fixed-size probabilistic representations makes it much easier to load balance expensive set operations. Simultaneously, ProbGraph achieves high accuracy of more than 90% for many inputs.

We also provide strong theoretical guarantees on work and depth of all the considered graph mining algorithms. Finally, we use our $|X \cap Y|$ estimators to derive novel estimators $\widehat{TC}$ on the triangle count TC in an arbitrary graph, achieving strong statistical properties such as MLE and exponential concentration quality bounds. *Our TC estimator based on MH has better theoretical properties (e.g., is MLE) than past theoretical results [46]–[54].*

Overall, PG enables trading a small amount of accuracy and storage for more performance. These tradeoffs are *tunable* by the user, who can select which aspect is most important.

II. Fundamental Concepts

We first present the used concepts ad symbols (see Table I).

A. Graph Model and Representation

A graph G is modeled as a tuple (V, E) with a set of vertices $(V, |V| = n)$ and edges $(E \subseteq V \times V, |E| = m)$. We model vertices with their integer IDs ($V = \{1, ..., n\}$). N_v and d_v denote the neighbors and the degree of a given vertex v ($N_v \subset V$); d is G's maximum degree. We store the input (i.e., not sketched) graph G using the standard Compressed Sparse Row (CSR) format, in which all neighborhoods N_v form a contiguous array ($2m$ words if G is undirected). Next, there is an array with n pointers to each representation of N_v. Each N_v is stored as a contiguous sorted array of vertex IDs.

B. Work-Depth Analysis of Parallel Algorithms

We use work-depth (WD) analysis for bounding run-times of parallel algorithms, to analyze PG set intersections (§ IV-F, Table IV), construction costs (§ VI-A, Table V), and graph algorithms (§ VI-B, Table VI). The *work* of an algorithm is the total number of operations and the *depth* is the longest sequential chain of execution in the algorithm (assuming infinite number of parallel threads executing the algorithm) [55], [56].

C. Set Algebra

When using arbitrary sets, we use symbols $X = \{x_1, ..., x_l\}$ and $Y = \{y_1, ..., y_l\}$. We use intersection ($X \cap Y$), union ($X \cup Y$), cardinality ($|X|$), and membership ($\in X$).

D. Probabilistic Set Representations

We consider Bloom filters (BF) and two variants of MinHash (MH). We pick different representations to better understand which ones are best suited for accelerating graph mining problems with high accuracy. All the proofs of theorems in the following sections are in the appendix. Figure 1 illustrates the representations considered.

Bloom filters The Bloom filter (BF) [35] is a space-efficient set representation that answers *membership queries* fast but with some probability of *false positives*. Formally, a Bloom filter $\mathcal{B}_X$ representing a set X consists of an l-element bit vector $\mathbf{B}_X$ (initialized to zeros) and b hash functions $h_1, \ldots, h_b$ (usually assumed to be independent) that map elements of X to integers in $[l]$ ($[l] \equiv \{1, ..., l\}$). The size of $\mathbf{B}_X$ is also denoted with B_X while the number of ones in $\mathbf{B}_X$ with $B_{X,1}$. Now, when constructing $\mathcal{B}_X$, for each element $x \in X$, one computes the corresponding hashes $h_1(x), \ldots, h_b(x)$. Then, the bits $\mathbf{B}_X[h_1(x)], ..., \mathbf{B}_X[h_b(x)]$ are set to one. Second, verifying if $x \in X$ is similar. First, all hash functions are evaluated for x. If all bits at the corresponding positions are set, i.e., $\forall i \in \{1, \ldots, b\} : \mathbf{B}_X[h_i(x)] = 1$, then x is considered to be in X. It is possible that some of these bits are set to one while adding other elements due to hash collisions, and x might be falsely reported as being an element of X. Minimizing the number of such *false positives* was addressed in many research works [57].

MinHash (k-Hash variant) MinHash (MH) [58] "sketches" a set X by hashing its elements to integers and keeping k elements with smallest hashes. An MH representation $\mathcal{M}_X$ of X consists of a set M_X with elements from X ($\forall_{x \in M_X}\ x \in X$) and k hash functions h_i, $i \in \{1, ..., k\}$. To construct $\mathcal{M}_X$, one computes all hashes $h_i(x)$ for each $x \in X$. Then, for *each* hash function h_i separately, one selects an element $x_{i,min} \in X$ that has the smallest hash $h_i(x)$. These elements form the final set $M_X = \{x_{1,min}, ..., x_{k,min}\}$. Note that M_X may be a multi set:

[1] with increasing sketch size, for a fixed input.

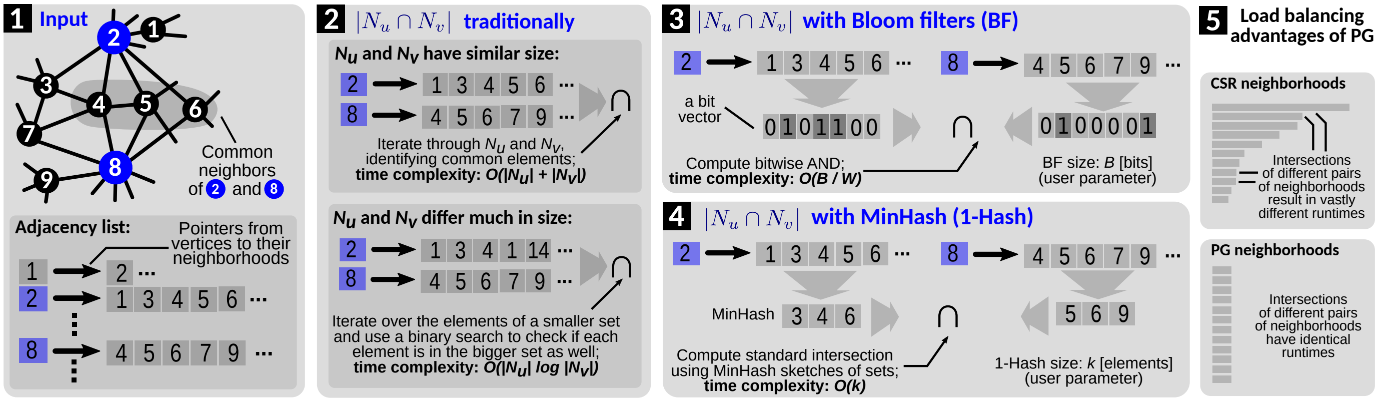

1 Input
Common neighbors of 2 and 8
Adjacency list:
Pointers from vertices to their neighborhoods
1 → 2 ···
2 → 1 3 4 5 6 ···
8 → 4 5 6 7 9 ···
2 $|N_u \cap N_v|$ traditionally
N_u and N_v have similar size:
2 → 1 3 4 5 6 ···
8 → 4 5 6 7 9 ···
∩
Iterate through N_u and N_v, identifying common elements;
time complexity: $O(|N_u| + |N_v|)$
N_u and N_v differ much in size:
2 → 1 3 4 1 14 ···
8 → 4 5 6 7 9 ···
∩
Iterate over the elements of a smaller set and use a binary search to check if each element is in the bigger set as well;
time complexity: $O(|N_u| \log |N_v|)$
3 $|N_u \cap N_v|$ with Bloom filters (BF)
2 → 1 3 4 5 6 ···
8 → 4 5 6 7 9 ···
a bit vector
0 1 0 1 1 0 0
∩
0 1 0 0 0 0 1
Compute bitwise AND;
time complexity: $O(B / W)$
BF size: B [bits] (user parameter)
4 $|N_u \cap N_v|$ with MinHash (1-Hash)
2 → 1 3 4 5 6 ···
8 → 4 5 6 7 9 ···
MinHash 3 4 6
∩
5 6 9
Compute standard intersection using MinHash sketches of sets;
time complexity: $O(k)$
1-Hash size: k [elements] (user parameter)
5 Load balancing advantages of PG
CSR neighborhoods
Intersections of different pairs of neighborhoods result in vastly different runtimes
PG neighborhoods
Intersections of different pairs of neighborhoods have identical runtimes

Fig. 1: Overview of selected PG set representations, how they are used to accelerate intersections of vertex neighborhoods, and in alleviating load imbalance. In panel "1", we show a part of an example input graph. In panel "2", we illustrate a traditional exact way to compute the count of shared neighbors $|N_u \cap N_v|$ of any vertices u and v. There are two variants of this operation: "merge" (the upper sub-panel), where one simply merges two sorted sets (it is more beneficial when sets have similar sizes), and "galloping" (the lower sub-panel), where – for each element from a smaller set, one uses binary search to check if this element is in the larger set (it works better when one set is much larger than the other one). In panel "3", we show how to compute $|N_u \cap N_v|$ with BF. Here, N_u and N_v are first converted into bit vectors (cf. wide vertical arrows). Then, the resulting bit vectors are intersected with a very fast bitwise AND operation (cf. wide horizontal arrows). In panel "4", we show how to compute $|N_u \cap N_v|$ with MH. Here, N_u and N_v are first appropriately hashed into much smaller subsets of N_u and N_v (cf. wide vertical arrows). The resulting sets can be intersected fast because they are much smaller than the original N_u and N_v (indicated with wide horizontal arrows). In panel "5", we show load balancing benefits from using PB (i.e., it is easy to load balance intersections of same-sized PG neighborhoods).

if $x_{i,min} = x_{j,min}$ for $i \neq j$, then M_X contains $x_{i,min}$ twice. MinHash was designed to approximate the Jaccard similarity index $J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$ that assesses the similarity of two sets X and Y. We have $J(X,Y) \in [0;1]$; $J(X,Y) = 0$ indicates no similarity while $J(X,Y) = 1$ means $X = Y$.

MinHash (1-Hash variant) The k-hash variant may be computationally expensive as it computes k different hash functions for all elements of X. To alleviate this, one can use a variant called 1-hash ($\mathcal{M}_X^1$) with only one hash function h. After computing hashes $h(x)$ for all $x \in X$, M_X^1 will contain k elements from X that resulted in k smallest hash values. Unlike k-hash, M_X^1 by definition never contains duplicates.

E. Estimators

A central concept in PG is an estimator: a rule for calculating an estimate of a given quantity based on observed data. We develop estimators of set sizes $\widehat{|X|}$ and set intersection cardinalities $\widehat{|X \cap Y|}$, and we use them to approximate $|N_v \cap N_u|$ for any two vertices u, v in considered graph algorithms.

We describe in more detail a specific BF estimator by Swamidass et al. [59]. We later generalize it and also prove novel bounds on its quality. Given a set X represented by a Bloom filter $\mathcal{B}_X$, one can estimate $|X|$ as

$$\widehat{|X|}_S = -\frac{B_X}{b}\log\left(1 - \frac{B_{X,1}}{B_X}\right). \qquad (1)$$

Graph, code	$G = (V, E)$	A graph G; V, E are sets of vertices and edges, respectively.
	n, m	Numbers of vertices and edges in G; $\lvert V\rvert = n, \lvert E\rvert = m$.
	d_v, N_v	The degree and neighbors of $v \in V$.
	$d, \bar{d}$	The maximum and the average degree in G ($\bar{d} = m/n$).
	TC	Count of triangles in a given graph.
	W	Size of a memory word [bits].
	s	Storage budget, i.e., space dedicated to PG structures.
BF	$\mathcal{B}_X, \mathbf{B}_X, \mathbf{B}_X[i]$	A Bloom filter; the associated bit vector; the i-th bit in $\mathbf{B}_X$.
	$B_X, B_{X,1}, B_{X,0}$	The size of $\mathbf{B}_X$ (#bits); number of ones and zeros in $\mathbf{B}_X$.
	b	The number of hash functions used with a given Bloom filter.
	h_i	An i-th associated hash function, $i \in \{1, ..., b\}$.
	p_f	The probability of a false positive of a given Bloom filter.
MinHash	$\mathcal{M}_X, M_X$	k-Hash variant; the approximating set based on input X.
	k	The number of elements stored in a given MinHash.
	h_i	An i-th associated hash function, $i \in \{1, ..., k\}$.
	$\mathcal{M}_X^1, M_X^1$	1-hash variant; the approximating set based on input X.

TABLE I: Important used symbols. "X" denotes the input set approximated by a respective representation (omitted if it is clear from context).

To derive this estimator, consider inserting a single element into a given BF $\mathcal{B}_X$. The probability that some single bit equals 0, in $\mathcal{B}_X$ with one hash function (that maps elements *uniformly* over the bit array), is $1 - 1/B_X$. When applying b hash functions (which follow the usual assumption of being independent), this probability is $(1 - 1/B_X)^b$. After inserting $|X|$ elements into $\mathcal{B}_X$, the probability that this one bit is 0 is $(1 - 1/B_X)^{b|X|} \approx \exp(-b|X|/B_X)$ (from the established identity for $e^{-1} \approx (1 - 1/x)^x$). Thus, the probability that this bit is set to 1, is $1 - (1 - 1/B_X)^{b|X|} \approx 1 - \exp(-b|X|/B_X)$. Then, observe that the number of bits set to 1 in $\mathcal{B}_X$, can be estimated as $B_{X,1} \approx B_X \cdot (1 - \exp(-b|X|/B_X))$ given the binomial density approximation used in Swamidass et al. [59]. When resolving this equation for $|X|$, we obtain Eq. (1).

F. Properties of Estimators

Figure 2 provides an intuitive description of the desirable statistical properties of PG. These properties enable highly accuracy empirical results (Section VIII) and attractive theoretical results for triangle count (Section VII).

All PG estimators are **asymptotically unbiased**. In such an estimator $\widehat{\theta}$, the difference between $\widehat{\theta}$'s expected value and the true value of the parameter being estimated θ converges to 0 for a fixed input and the size of the sketch that we use going to infinity (i.e., the bias of $\widehat{\theta}$ goes to 0, or, on average, $\widehat{\theta}$ hits θ when the sample size approaches the limit). Unbiased estimators are usually more desirable than biased ones: intuitively, they ensure zero average error (when estimating θ) after a given amount of trials. Next, each PG estimator of $\widehat{\theta}$ is also **consistent**, i.e, the sampling distribution of $\widehat{\theta}$ becomes increasingly more concentrated at θ with the increasing number of samples. Hence, if there are enough observations (in our case when the sketches are large enough), one can find θ with arbitrary precision. Asymptotic unbiasedness alone does not imply consistency; it requires also a vanishing variance (i.e., that the estimator variance converges to 0 with the increasing sample size).

We also verify if PG estimators are **maximum likelihood**. This class of estimators provides several powerful and useful properties, and is among the most important tools for estimation and inference in statistics [60]–[62]. Specifically, a maximum likelihood estimator (MLE) $\widehat{\theta}_{MLE}$ is an estimator that maximizes the *likelihood function* L, i.e., $\widehat{\theta}_{MLE} = \text{argmax}_{\theta \in \Theta}\ L(\mathbf{x}; \theta)$ (cf., Chapter 7 in [39]) where Θ is the parameter space. Here, the likelihood function L is defined as the probability of observing a given sample $\mathbf{x} = (x_1, ..., x_n)$ as a function of θ, i.e., $L \equiv P(X_1 = x_1, ..., X_n = x_n; \theta)$, where $X_1, ..., X_n$ represent a random sample from a given population. Thus, $\widehat{\theta}_{MLE}$ is the value of the parameter θ for which the observed sample is the most likely. This intuitive choice of an estimator leads under mild conditions to strong optimality properties such as consistency (discussed above), **invariance**, and **asymptotic efficiency**. An estimator $\widehat{\theta}$ is invariant if, whenever $\widehat{\theta}$ is the MLE of θ, then for any function $\tau(\cdot)$, $\tau(\widehat{\theta})$ is the MLE of $\tau(\theta)$. This is useful if complicated functions of the parameter θ are of interest since $\tau(\widehat{\theta}_{MLE})$ inherits automatically all the properties of $\widehat{\theta}_{MLE}$. Finally, the asymptotic efficiency certifies that $\widehat{\theta}_{MLE}$ attains, under mild regularity conditions, the Cramer-Rao bound [39]. To understand the importance of this result, we need to introduce the Mean Squared Error (MSE) of an estimator. MSE measures the average squared difference between the estimator and the parameter, i.e., $MSE(\widehat{\theta}, \theta) = E_\theta[(\widehat{\theta} - \theta)^2]$). Thus, the asymptotic efficiency implies that there exists no consistent estimator of θ that achieves a lower MSE than $\widehat{\theta}_{MLE}$. Next, given the well known decomposition of the MSE into bias squared plus variance (i.e., $MSE(\widehat{\theta}) = Bias(\widehat{\theta}, \theta)^2 + Var_\theta(\widehat{\theta})$), we can also conclude that among all the consistent estimators of θ, no one has a strictly smaller variance than $\widehat{\theta}_{MLE}$. Another interesting implication of this result is the **asymptotic normality** of MLE. This property allows to build valid *confidence intervals* for the parameter of interest based on the normal distribution (asymptotically in sketch size, for a fixed input).

G. Concentration Bounds

We use the notion of a *concentration inequality*. Overall, such an inequality *bounds the deviation of a given random variable X from some value* (usually the expectation $E[X]$). In this work, we mainly use the Chebyshev [38], Hoeffding [63] and Serfling inequalities [64].

III. Sets & $|X \cap Y|$ In Graph Algorithms

We first identify algorithms that use $|X \cap Y|$. A graph itself can be modeled as a collection of sets: each vertex neighborhood N_v is essentially a set. In PG, we use this observation to approximate the graph structure and operations by using probabilistic set representations in place of N_v and $|N_v \cap N_u|$, for any vertices u and v. In the following listings, the "X" and "Y" general symbols are replaced with specific sets. Operations approximated by PG are marked with the blue color. "`[in par]`" means that a loop is parallelized. We ensure that the parallelization does not involve conflicting memory accesses. For clarity, we focus on *formulations* and we discuss parallelization details in Section VI.

In **Triangle Counting (TC)** [65], [66] (Listing 1), one counts the total number of triangles tc in an undirected graph. An example application of TC is computing clustering coefficients [67]. For each vertex u, one computes the cardinalities of the intersections of N_u, the set of neighbors of u, with the sets of the neighbors of each neighbor of u (Lines 4-7).

```
1 /* Input: A graph $G$. Output: Triangle count $tc \in \mathbb{N}$. */
2 //Derive a vertex order $R$ s.t. if $R(v) < R(u)$ then $d_v \leq d_u$:
3 for $v \in V$ [in par] do: $N_v^+ = \{u \in N_v | R(v) < R(u)\}$
4 $tc$ = 0; //Init $tc$; for all neighbor pairs, increase $tc$:
5 //Now, derive the actual count of triangles:
6 $v \in V$ [in par] do:
7   for $u \in N_v^+$ [in par] do: $tc$ += $| N_v^+ \cap N_u^+ |$
```

Listing 1: Triangle Counting (Node Iterator).

We also consider higher-order **Clique Counting**, a problem important for dense subgraph discovery [68]. Listing 2 contains 4-clique listing. We reformulated the original scheme [68] (without changing its time complexity) to expose $|X \cap Y|$. The algorithm presented generalizes TC.

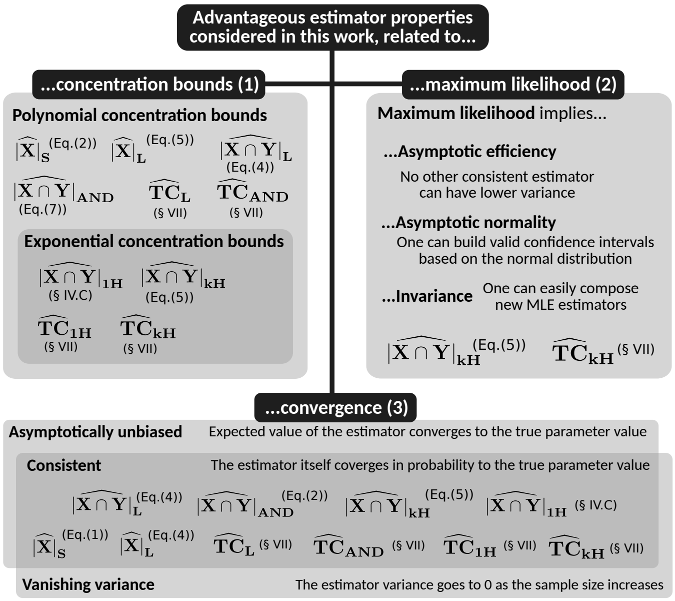

Advantageous estimator properties considered in this work, related to...
...concentration bounds (1)
...maximum likelihood (2)
Polynomial concentration bounds
$\widehat{|\mathbf{X}|}_{\mathbf{S}}$ (Eq.(2))
$\widehat{|\mathbf{X}|}_{\mathbf{L}}$ (Eq.(5))
$\widehat{|\mathbf{X} \cap \mathbf{Y}|}_{\mathbf{L}}$ (Eq.(4))
$\widehat{|\mathbf{X} \cap \mathbf{Y}|}_{\mathbf{AND}}$ (Eq.(7))
$\widehat{\mathbf{TC}}_{\mathbf{L}}$ (§ VII)
$\widehat{\mathbf{TC}}_{\mathbf{AND}}$ (§ VII)
Exponential concentration bounds
$\widehat{|\mathbf{X} \cap \mathbf{Y}|}_{\mathbf{1H}}$ (§ IV.C)
$\widehat{|\mathbf{X} \cap \mathbf{Y}|}_{\mathbf{kH}}$ (Eq.(5))
$\widehat{\mathbf{TC}}_{\mathbf{1H}}$ (§ VII)
$\widehat{\mathbf{TC}}_{\mathbf{kH}}$ (§ VII)
Maximum likelihood implies...
...Asymptotic efficiency
No other consistent estimator can have lower variance
...Asymptotic normality
One can build valid confidence intervals based on the normal distribution
...Invariance
One can easily compose new MLE estimators
$\widehat{|\mathbf{X} \cap \mathbf{Y}|}_{\mathbf{kH}}$ (Eq.(5))
$\widehat{\mathbf{TC}}_{\mathbf{kH}}$ (§ VII)
...convergence (3)
Asymptotically unbiased
Expected value of the estimator converges to the true parameter value
Consistent
The estimator itself coverges in probability to the true parameter value
$\widehat{|\mathbf{X} \cap \mathbf{Y}|}_{\mathbf{L}}$ (Eq.(4))
$\widehat{|\mathbf{X} \cap \mathbf{Y}|}_{\mathbf{AND}}$ (Eq.(2))
$\widehat{|\mathbf{X} \cap \mathbf{Y}|}_{\mathbf{kH}}$ (Eq.(5))
$\widehat{|\mathbf{X} \cap \mathbf{Y}|}_{\mathbf{1H}}$ (§ IV.C)
$\widehat{|\mathbf{X}|}_{\mathbf{S}}$ (Eq.(1))
$\widehat{|\mathbf{X}|}_{\mathbf{L}}$ (Eq.(4))
$\widehat{\mathbf{TC}}_{\mathbf{L}}$ (§ VII)
$\widehat{\mathbf{TC}}_{\mathbf{AND}}$ (§ VII)
$\widehat{\mathbf{TC}}_{\mathbf{1H}}$ (§ VII)
$\widehat{\mathbf{TC}}_{\mathbf{kH}}$ (§ VII)
Vanishing variance
The estimator variance goes to 0 as the sample size increases

Fig. 2: Considered estimator properties and the associated PG estimators. The advantageous properties of ProbGraph estimators belong to three classes of properties: **(1) having concentration bounds**, **(2) being maximum likelihood**, and **(3) convergence**. **Intuitively**, "(1)" means that we can bound the probability that a PG estimator deviates from the true parameter value for every sketch size. "(2)" means that a PG estimator identifies the parameter value which is more likely to have produced the data we observed. "(3)" means that a PG estimator converges to the true parameter value as the sketch size increases. In the figure, we list different desirable properties implied by (1), (2), and (3). Formal definitions of the considered properties can be found in Section II.

```
/* Input: A graph $G$. Output: Number of 4-cliques $ck \in \mathbb{N}$. */
/Derive a vertex order $R$ s.t. if $R(v) < R(u)$ then $d_v \le d_u$:
for $v \in V$ [in par] do: $N_v^+ = \{u \in N_v | R(v) < R(u)\}$
$ck$ = 0;
for $u \in V$ [in par] do:
  for $v \in N_u^+$ [in par] do:
    $C_3$ = $N_u^+ \cap N_v^+$ //Find 3-cliques
    for $w \in C_3$ do: //For each 3-clique...
      $ck$ += $|N_w^+ \cap C_3|$ //Find 4-cliques
```

Listing 2: Reformulated 4-Clique Counting.

Vertex Similarity measures, used in graph databases and others [69]–[73], assess how similar two vertices v and u are, see Listing 3. They can be used on their own, or as a building block of more complex algorithms such as clustering. Many of these schemes use the cardinality of set intersection. This includes Jaccard, Common Neighbors, Total Neighbors, or Adamic Adar. Vertex Similarity is the basic building block of **Link Prediction** and Clustering.

```
/* Input: A graph $G$. Output: Similarity $S \in \mathbb{R}$ of sets $A, B$.
* Most often, $A$ and $B$ are neighborhoods $N_u$ and $N_v$
 * of vertices $u$ and $v$. */
//Jaccard similarity:
$S_J(A,B)$ = $|A \cap B|$ / $|A \cup B|$ = $|A \cap B|$ / ($|A|$ + $|B|$ - $|A \cap B|$)
//Overlap similarity:
$S_O(A,B)$ = $|A \cap B|$ / min($|A|$, $|B|$)
//Certain measures are only defined for neighborhoods:
$S_A(v,u)$ = $\sum_w (1/\log |N_w|)$ //where $w \in$ $N_v \cap N_u$; Adamic Adar
$S_R(v,u)$ = $\sum_w (1/|N_w|)$ //where $w \in$ $N_v \cap N_u$; Resource Alloc.
$S_C(v,u)$ = $|N_v \cap N_u|$ //Common Neighbors
$S_T(v,u)$ = $|N_v \cup N_u|$ = $|N_v|$ + $|N_u|$ - $|N_v \cap N_u|$ //Total Neighbors
```

Listing 3: Example vertex similarity measures [74].

Graph Clustering is a broadly studied problem [42]. Listing 4 shows Jarvis-Patrick clustering [44], a scheme that uses vertex similarity to determine whether these two vertices are in the same cluster, and relies heavily on $|X \cap Y|$.

```
/* Input: A graph $G = (V, E)$. Output: Clustering $C \subseteq E$
 * of a given prediction scheme. */
//Use a similarity $S_C(v,u)$ = $|N_v \cap N_u|$ (see Listing 3).
for $e = (v,u) \in E$ [in par] do: //$\tau$ is a user-defined threshold
  if $|N_v \cap N_u|$ $> \tau$: $C$ ∪= $\{e\}$
//Other clustering schemes use other similarity measures.
```

Listing 4: Jarvis-Patrick clustering.

Link Prediction There are many schemes for predicting whether two non-adjacent vertices can become connected in the future in the context of evolving networks [75]. Assessing the accuracy of a specific link prediction scheme S is done with a simple algorithm [76] shown in Listing 5. We start with some graph with *known* links (edges). We derive $E_{sparse} \subseteq E$, which is E with random links removed; $E_{sparse} = E \setminus E_{rndm}$. $E_{rndm} \subseteq E$ are randomly selected *missing* links from E (*links to be predicted*). We have $E_{sparse} \cup E_{rndm} = E$ and $E_{sparse} \cap E_{rndm} = \emptyset$. Now, we apply the link prediction scheme S (that we want to test) to each edge $e \in (V \times V) \setminus E_{sparse}$. The higher a value $S(e)$, the more probable e is to appear in the future (according to S). Now, the effectiveness ef of S is computed by verifying how many of the edges with highest prediction scores ($E_{predict}$) actually are present in the original dataset E: $ef = |E_{predict} \cap E_{rndm}|$.

A. Real-World Applications

Graph problems targeted by ProbGraph have numerous real-world applications because the underlying operation $|X \cap Y|$, used to find the counts of the shared neighbors, is a common

```
/* Input: A graph $G = (V, E)$. Output: Effectiveness $ef$
 * of a given prediction scheme. */
$E_{rndm}$ = /* Random subset of $E$ */
$E_{sparse}$ = $E \setminus E_{rndm}$ /* Edges in $E$ after removing $E_{rndm}$ */
//For each $e \in (V \times V) \setminus E_{sparse}$, derive score $S(e)$ that
//determines the chance that $e$ appears in future. Here,
//one can use any vertex similarity scheme $S$.
for $e = (v, u) \in (V \times V) \setminus E_{sparse}$ [in par] do: compute $S(v, u)$
$E_{predict}$ = /* Pick selected top edges with highest $S$ scores.*/
$ef$ = $|E_{predict} \cap E_{rndm}|$ //Derive the effectiveness.
```

Listing 5: Link prediction testing.

building block in many real-world problems in domains ranging from network science or sociology, through chemistry or biology, to the Internet studies [42].

Triangle counting is used to obtain the *network cohesion*, an important measure of connectedness and “togetherness” of a group of vertices [77], [78]. Specifically, for any subgraph $S \subseteq V$, S’s cohesion is $TC[S]/\binom{|S|}{3}$, where $TC[S]$ is the triangle count of S; note that S may also form V (in which case we obtain the cohesion of the whole graph). Another example is discovering communities [79], [80], by computing the clustering coefficient defined as $3 \cdot TC[S]/\binom{|S|}{3}$ Other use cases include *spam detection* (standard and spam sites differ in the respective counts of triangles that they belong to), optimization of query planning in databases [81], uncovering hidden thematic layers in WWW [82], or studying differences between gene interactomes of various species [83].

The considered Jarvis-Patrick clustering can be used in adaptive web search based on automatic construction of user profiles. A critical step in this use case is generation of clusters of users, which is directly achieved using the clustering scheme addressed in PG [84]. Other selected examples are drug design (by predicting plasma protein bindings [85]), screening and generating overviews of chemical databases (by computing clusters of related molecules) [86], or analyzing single-cell RNA sequences (by approximating smooth low-dimensional surfaces that model states of cells) [87], [88].

Other considered problems also have numerous applications. In short, clique counting is used in social network analysis (cf. the established textbooks [89, Chapter 11] and [90, Chapter 2]) to find large and dense network regions [91]–[95] or in topological approaches to network analysis [96]. Link prediction and vertex similarity are used throughout the whole graph data mining in many parts of network science and others, as illustrated in numerous surveys and textbooks [4], [11], [12], [75], [97]–[99].

IV. Approximating $|X \cap Y|$

We now show how to derive approximate set intersection *cardinality* $|X \cap Y|$ both *fast* and *with high accuracy*. We provide selected results for BF and MH (less competitive outcomes are in the Appendix). In this section, we assume arbitrary sets X and Y, to ensure that our outcomes are of interest beyond graph mining. From Section VII onwards, we focus on graph mining by applying the results from this section to $|\widehat{N_u \cap N_v}|$.

A. Section Overview and Intuition

We first outline the section structure and provide the intuition behind the key parts. We first present estimators for $|X \cap Y|$, designed using BF (§ IV-B), k-Hash (§ IV-C), and 1-Hash (§ IV-D). Then, we compare the obtained estimators regarding their accuracy (§ IV-E) and the amount of parallelism (§ IV-F).

We provide concentration bounds for all the estimators. We present here selected ones, the others are in the supplementary material together with the proofs of all the propositions of this section. A generic form of a concentration bound from this section is $P(|\text{estimator} - \text{true\_value}| \geq t) \leq f(t)$. Intuitively, this means that we can bound (i.e., by the function $f(t)$) the probability that a PG estimator deviates (i.e., more than t) from the true parameter value for every sketch size. The function $f(t)$ (either polynomial or exponential in t for all PG estimators) determines the speed at which a given PG estimator concentrates around the true parameter value.

B. Approximating $|X \cap Y|$ with Bloom Filters

We introduce a new estimator $|\widehat{X \cap Y}|_{AND}$ and we give a bound on its accuracy. Specifically, for two sets X and Y represented by $\mathcal{B}_X$ and $\mathcal{B}_Y$, we apply the estimator from Eq. (1) to $\mathcal{B}_{X \cap Y}$, obtaining

$$|\widehat{X \cap Y}|_{AND} = -\frac{B_{X \cap Y}}{b} \log\left(1 - \frac{B_{X \cap Y,1}}{B_{X \cap Y}}\right) \tag{2}$$

where $B_{X \cap Y} = B_X = B_Y$ is the BF size (cf. Table I). Next, we prove an important property of $|\widehat{X \cap Y}|_{AND}$. Note that the following property also holds for the estimator by Swamidass [59] from Eq. (1).

Proposition IV.1. *Let $|\widehat{X \cap Y}|_{AND}$ be the estimator defined in Eq. (2). For $B_{X \cap Y}, b \in \mathbb{N}$ such that $b = o(\sqrt{B_{X \cap Y}})$, and a set $X \cap Y$ such that $b|X \cap Y| \leq 0.499 B_{X \cap Y} \cdot \log B_{X \cap Y}$ the following holds:*

$$E\left[\left(|\widehat{X \cap Y}|_{AND} - |X \cap Y|\right)^2\right] \leq (1 + o(1)) \left(e^{|X \cap Y| b/(B_{X \cap Y} - 1)} \frac{B_{X \cap Y}}{b^2} - \frac{B_{X \cap Y}}{b^2} - \frac{|X \cap Y|}{b}\right)$$

Overall, Proposition IV.1 shows that we can bound the mean squared error (MSE) of $|\widehat{X \cap Y}|_{AND}$ (and also $\widehat{|X|}_S$ from Eq. (1)). By Chebyshev’s inequality[2], we obtain the following concentration result:

$$P\left(\left||\widehat{X \cap Y}|_{AND} - |X \cap Y|\right| \geq t\right) \leq (1 + o(1)) \frac{\left(e^{|X \cap Y| b/(B_{X \cap Y} - 1)} \frac{B_{X \cap Y}}{b^2} - \frac{B_{X \cap Y}}{b^2} - \frac{|X \cap Y|}{b}\right)}{t^2} \tag{3}$$

[2] We apply the inequality on the MSE to derive a bound for $P\left(\left||\widehat{X \cap Y}|_{AND} - |X \cap Y|\right| \geq t\right)$ rather than for $P\left(\left||\widehat{X \cap Y}|_{AND} - E(|\widehat{X \cap Y}|_{AND})\right| \geq t\right)$ (as is usually done).

We can strengthen the intuition on the behavior of $|\widehat{X \cap Y}|_{AND}$ by taking the limit for $B_{X\cap Y} \to \infty$ in Eq. (2). We call $|\widehat{X \cap Y}|_{L}$ this limiting estimator:

$$|\widehat{X \cap Y}|_{L} \equiv \lim_{B_{X\cap Y} \to \infty} |\widehat{X \cap Y}|_{AND} = \frac{B_{X\cap Y,1}}{b} \tag{4}$$

Hence, as $B_{X\cap Y}$ increases, $|\widehat{X \cap Y}|_{AND}$ *rescales the number of ones in the BF* by $\frac{1}{b}$ because $|\widehat{X \cap Y}|_{AND} \sim \frac{B_{X\cap Y,1}}{b}$ for X, Y, b fixed and $B_{X\cap Y} \to \infty$. In Section VIII, we will show that – depending on the choice of the scaling factor $\frac{1}{b}$ which impacts the bias-variance trade-off – there are cases where $|\widehat{X \cap Y}|_{L}$ is better than $|\widehat{X \cap Y}|_{AND}$.

Note that $|\widehat{X \cap Y}|_{AND}$ uses the count of ones in a BF $\mathbf{B}_{X\cap Y}$. This number cannot be computed from individual BFs $\mathbf{B}_{X}$ and $\mathbf{B}_{Y}$. In practice, we use $\mathbf{B}_{X\cap Y} \approx \mathbf{B}_{X}$ AND $\mathbf{B}_{Y}$ (where "AND" indicates a logical bitwise AND operation) and use the result of AND to obtain $B_{X\cap Y,1}$. This may somewhat increase the false positive probability, but – as the results in Section VIII show – does not prevent high accuracy.

C. Approximating $|X \cap Y|$ with k-Hash

To estimate $|X \cap Y|$ with MinHash, one first uses the definition of the Jaccard similarity index $J_{X,Y} = |X \cap Y|/|X \cup Y|$ (cf. § II-D) and, together with the well-known set algebraic expression $|X \cup Y| = |X| + |Y| - |X \cap Y|$, rewrites it to obtain the following estimator:

$$|\widehat{X \cap Y}|_{kH} = \frac{\widehat{J_{X,Y}}_{kH}}{1 + \widehat{J_{X,Y}}_{kH}} (|X| + |Y|) \tag{5}$$

where $\widehat{J_{X,Y}}_{kH} = \frac{|M_X \cap M_Y|}{k}$ is itself an unbiased estimator of $J_{X,Y}$ (see [100] for a proof). If we assume that the k hash functions are independent and perfectly random (a usual assumption[3]), we have $|M_X \cap M_Y| \sim Bin(k, J_{X,Y})$, i.e., $|M_X \cap M_Y|$ follows the binomial distribution, where the number of trials equals the number of hash functions k and the probability of success is the true Jaccard coefficient (this is valid by the construction of k-Hash, see [101]). Thus, we can derive the expectation and the variance of $|\widehat{X \cap Y}|_{kH}$ adapting the formulas for the moments of a binomial random variable (provided in the Appendix).

We develop the following concentration bound (this is the first exponential bound for $|\widehat{X \cap Y}|_{kH}$):

Proposition IV.2. *Let $|\widehat{X \cap Y}|_{kH}$ be the estimator from Eq. (5). Then, an upper bound for the probability of deviation from the true $|X \cap Y|$, at a given distance $t \geq 0$, is:*

$$P\left(\left| |\widehat{X \cap Y}|_{kH} - |X \cap Y| \right| \geq t\right) \leq 2e^{-\frac{2\ k\ t^2}{(|X|+|Y|)^2}} \tag{6}$$

We stress that $|\widehat{X \cap Y}|_{kH}$ derived with k-hash can also be interpreted as a *maximum likelihood estimator (MLE)* (cf. § II-E) for $|X \cap Y|$ because of the invariance property outlined in § II-F (details are provided together with the proof). Thus, our estimator inherits all the properties of MLE such as consistency and asymptotic efficiency. Moreover, the bound is *exponential*, i.e., the distance between the estimator and the true value of $|X \cap Y|$ decreases exponentially. Finally, we stress that $|\widehat{X \cap Y}|_{kH}$ *is asymptotically efficient*, i.e., no other estimator (using only this sketch) can have lower variance (for a fixed input and asymptotically for $k \to \infty$).

D. Approximating $|X \cap Y|$ with 1-Hash

The 1-Hash estimator is similar to k-Hash in that we first estimate the Jaccard similarity itself as $\widehat{J_{X,Y}}_{1H} = \frac{|M_X^1 \cap M_Y^1|}{k}$. Similarly to the estimator used in k-Hash, this is itself an unbiased estimator of $J_{X,Y}$. Then, we use it to estimate $|X \cap Y|$: $|\widehat{X \cap Y}|_{1H} = \frac{\widehat{J_{X,Y}}_{1H}}{1+\widehat{J_{X,Y}}_{1H}} (|X| + |Y|)$.

Recall that the 1-Hash representation of X differs qualitatively from the k-Hash variant in that (1) M_X^1 does not contain duplicates, and (2) $\mathcal{M}_X^1$ uses only one hash function. The k elements maintained in a 1-Hash are not independent, as we are in a *sampling without replacement* scheme[4]. This also means that k-Hash can have duplicates, which is not possible with 1-Hash. Thus, $|M_X^1 \cap M_Y^1| \sim Hypergeometric(|X \cup Y|, |X \cap Y|, k)$ where $|X \cup Y|$ is the population size, $|X \cap Y|$ is the number of success states in the population, and k is the number of draws. This implies that we can derive the expectation and the variance of $|\widehat{X \cap Y}|_{1H}$ by adapting the formulas for the moments of an hypergeometric random variable. We provide the formulas for the expectation in the Appendix. We now provide the same concentration bound as in the case of k-Hash.

Proposition IV.3. *Consider $|\widehat{X \cap Y}|_{1H}$. Then, an upper bound for the probability of deviation from the true intersection set size, at a given distance $t \geq 0$, is:*

$$P\left(\left| |\widehat{X \cap Y}|_{1H} - |X \cap Y| \right| \geq t\right) \leq 2e^{-\frac{2\ k\ t^2}{(|X|+|Y|)^2}} \tag{7}$$

The bound suggests that 1-Hash can be better in practice than k-Hash. They both have exponential bounds but 1-Hash requires hashing elements using only *one* hash function. Thus, it is faster to compute.

E. Analysis of Accuracy of $|\widehat{X \cap Y}|$

We summarize our theory developments into estimating $|X \cap Y|$ in Table II (estimators) and Table III (bounds). These results are also applicable to general estimators of $|X|$ (cf. § II-E) and we also show them in the table. We provide deviation bounds for all PG estimators. The estimator for k-hash is an MLE. Moreover, the k-Hash and 1-Hash bounds are exponential. This means that the estimates are unlikely to deviate much from the true value.

[3] To satisfy this assumption, one could just store perfectly random permutations on the set of vertices without increasing asymptotic complexity.

[4] Contrarily, k-Hash is a *sampling with replacement scheme* and explains why $|M_X \cap M_Y| \sim Bin(k, J_{X,Y})$ for k-Hash

Result	Where	Class	AU	CN	ML	IN	AE
$\widehat{\lvert X\rvert}_S$	Eq. (1)	BF	👍★	👍★	✗	✗	✗
$\lvert\widehat{X \cap Y}\rvert_{AND}$ ★	Eq. (2)	BF	👍★	👍★	✗	✗	✗
$\lvert\widehat{X \cap Y}\rvert_{L}$ ★	§ IV-B	BF	👍★	👍★	✗	✗	✗
$\lvert\widehat{X \cap Y}\rvert_{kH}$	Eq. (5)	k-Hash	👍★	👍★	👍★	👍★	👍★
$\lvert\widehat{X \cap Y}\rvert_{1H}$	§ IV-D	1-Hash	👍★	👍★	✗	✗	✗

TABLE II: Summary of theoretical results (estimators) related to $\widehat{|X|}$ and $|\widehat{X \cap Y}|$. "★": a new result provided in this work (a new estimator or proving a certain novel property of a given estimator). "**CN**": a consistent estimator. "**AU**": an asymptotically unbiased estimator. "**ML**": an MLE estimator. "**IN**": an invariant estimator. "**AE**": an asymptotically efficient estimator.

Result	Where	Class	Q	MS	CO
$\widehat{\lvert X\rvert}_S$ ★	Eq. (1)	BF	P ★	👍	👍
$\lvert\widehat{X \cap Y}\rvert_{AND}$ ★	Eq. (3)	BF	P ★	👍	👍
$\lvert\widehat{X \cap Y}\rvert_{L}$ ★	§ IV-B	BF	P ★	👍	👍
$\lvert\widehat{X \cap Y}\rvert_{kH}$ ★	Eq. (6)	k-Hash	E ★	✗	👍
$\lvert\widehat{X \cap Y}\rvert_{1H}$ ★	Eq. (7)	1-Hash	E ★	✗	👍

TABLE III: Summary of theoretical results (bounds) related to $\widehat{|X|}$ and $|\widehat{X \cap Y}|$. "★": a new result provided in this work. "**Q**": the quality of a given bound, "**P**": polynomial, "**E**": exponential. "**MS**": an MSE bound. "**CO**": a concentration bound.

F. Analysis of Parallelism in $|\widehat{X \cap Y}|$

In Table IV, we provide a work-depth analysis of parallelism in different estimators, when applied to intersecting vertex neighborhoods. For the exact intersection applied to CSR, we use two variants: merge (more advantageous when neighborhoods are similar in size) and galloping (used when neighborhoods vary in size by a large factor); the exact schemes and work/depth are provided in numerous works [1], [4], [65]. *Importantly, using PG gives asymptotic advantages in both work and depth over CSR*. Work and depth in BF are dominated by – respectively – the bitwise AND over participating bit vectors (taking $O(B_X/W)$ work) and the final sum of 1s over the resulting bitvector (taking $O(\log B_X/W)$ depth using a simple binary tree reduction). Note that B_X is always expressed in bits and thus we divide it with the SIMD width (or plain memory word size) W to obtain the actual counts of operations. MH representations are series of up to k vertex IDs and thus they use standard intersections. Both BF and MH based intersection have attractive work and depth as $\log k$ and $\log(B_X/W)$ are in practice very small.

	CSR (merge)	**CSR** (gallop.)	**BF**	k**–Hash**	**1–Hash**
Work:	$O(d_u + d_v)$	$O(d_u \log d_v)$	$O\left(\frac{B_X}{W}\right)$	$O(k)$	$O(k)$
Depth:	$O(\log(d_u + d_v))$	$O(\log(d_u + d_v))$	$O\left(\log\left(\frac{B_X}{W}\right)\right)$	$O(\log k)$	$O(\log k)$

TABLE IV: Work and depth of simple parallel algorithms for deriving $|N_u \cap N_v|$ (cardinality of the result of intersecting neighborhoods of vertices u, v).

V. Using ProbGraph with Graph Algorithms

We carefully design and implement PG as an easy-to-use library offering different set representations. To use PG, the user (1) creates selected probabilistic representations of vertex neighborhoods (BF, k-Hash, or 1-Hash), (2) plugs in PG routines for obtaining $|\widehat{X \cap Y}|$ in place of the exact set intersections. For example, to use PG with graph algorithms from Section III, one replaces the operations indicated with blue color with PG routines. As an example, in Listing 6, we compare obtaining Jaccard similarity of two neighborhoods with an exact scheme and with a PG routine based on BF. We ensure that one can flexibly select an arbitrary estimator $|\widehat{X \cap Y}|$ because – as our evaluation (Section VIII) shows – no single representation works best in all cases.

```
//Input: Graph G, two vertices u and v
//Create a standard CSR graph with G as the input graph
CSRGraph g = CSRGraph(G);
//Create a ProbGraph representation of G based on Bloom filters
ProbGraph pg = ProbGraph(g, BF, 0.25); //Use the 25% storage budget

//Derive the exact intersection cardinality |N_u ∩ N_v|
int interEX = pg.int_card(g.N(u), g.N(v));
//Derive the estimator |N_u ∩ N_v|_AND
int interBF = pg.int_BF_AND(pg.N(u), pg.N(v));

//Compute the exact Jaccard coefficient between u and v
double jacEX = interEX / (g.N(u).size() + g.N(v).size() - interEX)
//Compute the approximate Jaccard coefficient based on BF
double jacBF = interBF / (g.N(u).size() + g.N(v).size() - interPG)
```

Listing 6: Obtaining exact and approximate Jaccard (see Listing 3)

A. Tradeoffs Between Storage, Accuracy, & Performance

Each probabilistic set representation considered in PG offers a tradeoff between performance, storage, and accuracy. In general, the smaller a representation is, the faster to process it becomes and the less storage it needs, but also the less accurate it becomes. To control this tradeoff, we introduce a generic parameter s that enables explicit control of the storage budget. $s \in [0; 1]$ specifies how much additional memory (on top of the storage needed for the default CSR graph representation) is needed to maintain the PG estimators. In evaluation, we do not exceed more than 33% of the additional needed storage.

VI. Design & Implementation

Each BF is implemented as a simple bit vector. $|\widehat{X \cap Y}|$ can then be computed using bitwise AND over X and Y, with Eq. (2). Computing $B_{X \cap Y}$ can easily be parallelized and accelerated with vectorization [102]: the problem is embarrassingly parallel and the bitwise AND is supported with SIMD technologies such as AVX deployed in Intel CPUs, GPUs, and others. We also use the `popcnt` CPU instructions [103] to speed up deriving the number of ones in a bit vector (1-bits), needed to obtain $B_{X \cap Y,1}$ in Eq. (2); `popcnt` counts the number of 1-bits in one memory word *in one CPU cycle*.

1–Hash and k–Hash are both series of integers. The estimators for $|X \cap Y|$ based on 1–Hash and k–Hash are dominated by intersecting sets of k numbers. As $k \ll d$, it is much faster than the corresponding operations on exact neighborhoods.

A. Parallel Construction

Table V provides work and depth of constructing all probabilistic set representations used in PG. As with the intersection computation, the construction process is also parallelizable, exhibiting very low depth. During evaluation (Section VIII), we show that it also does not pose a bottleneck in practice.

Representation of N_v	**‡ Size** [bits]	**Construction (work)**	**Construction (depth)**
BF	B_X	$O(bd_v)$	$O(\log(bd_v))$
k-Hash	Wk	$O(kd_v)$	$O(\log d_v)$
1-Hash	Wk	$O(d_v)$	$O(\log d_v)$

TABLE V: Work/depth of simple algorithms for constructing a probabilistic PG set representation of a given neighborhood N_v. In BF, one must iterate over all b hash functions and all d_v neighbors, thus the work is dominated by bd_v (cf. § II-D). All the hash function evaluations can run in parallel, but – in the worse case – they may write to the same cell in the BF bit vector, giving depth $O(\log(bd_v))$ (parallelization with a binary tree reduction). Derivations for MH are similar; the work and depth are dominated by evaluations of hash functions and by finding k smallest elements among d_v ones, respectively.

	CSR	**PG (BF)**	**PG (MH)**
Triangle Counting (work):	$O\left(nd^2\right)$	$O\left(\frac{ndB_X}{W}\right)$	$O\left(ndk\right)$
Triangle Counting (depth):	$O\left(\log d\right)$	$O\left(\log\left(\frac{B_X}{W}\right)\right)$	$O\left(\log k\right)$
4-Clique Counting (work):	$O\left(nd^3\right)$	$O\left(\frac{nd^2B_X}{W}\right)$	$O\left(nd^2k\right)$
4-Clique Counting (depth):	$O\left(\log^2 d\right)$	$O\left(\log d \log\left(\frac{B_X}{W}\right)\right)$	$O\left(\log^2 k\right)$
Clustering (work):	$O\left(nd^2\right)$	$O\left(\frac{ndB_X}{W}\right)$	$O\left(ndk\right)$
Clustering (depth):	$O\left(\log d\right)$	$O\left(\log\left(\frac{B_X}{W}\right)\right)$	$O\left(\log k\right)$
Vertex sim. (work):	$O\left(d^2\right)$	$O\left(\frac{B_X}{W}\right)$	$O\left(k\right)$
Vertex vim. (depth):	$O\left(\log d\right)$	$O\left(\log\left(\frac{B_X}{W}\right)\right)$	$O\left(\log k\right)$

TABLE VI: Advantages of ProbGraph in work and depth over exact baselines.

B. Parallelism in ProbGraph-Enhanced Graph Algorithms

Parallelization of graph algorithms enhanced with ProbGraph is straightforward and is based on the listings from Section III. Specifically, all the loops marked with [in par] can be executed in parallel. Then, all the instances of set intersection cardinality are executed using a user-specified PG estimator. The parallel execution of these estimators (cf. § IV-F and Table IV) enables better work and depth of graph mining algorithms than with the default CSR implementation. We illustrate this in Table VI. Here, work and depth of CSR based routines are standard results known from extensive works in parallel algorithm design [7], [9], [56], [65], [104]. For example, in TC, the two outermost loops can be executed fully in parallel, and the nested set intersection dominates depth (d is the maximum degree in a graph). Both work and depth for PG baselines are derived by replacing the nested exact $|X \cap Y|$ operation with the corresponding PG schemes and results from Table IV. These asymptotic advantages are supported with empirical outcomes detailed in Section VIII.

C. Implementation Details and Infrastructure

We use the GMS platform [4], a high-performance parallel graph mining infrastructure, for implementing the baselines. Loading graphs from disk and building the CSR representations is done with the GAP Benchmark Suite [105]. We use the MurmurHash3 hash function [106], well-known for its speed and simplicity. We use the current time in milliseconds as a random seed. For parallelization, we use OpenMP [107]. The whole implementation is available online.[5]

VII. Theoretical Analysis of Accuracy

We now illustrate that ProbGraph enables obtaining not only strong theoretical accuracy guarantees on the set intersection cardinality, but also on graph properties. As an example, we now use our estimators $\widehat{|X \cap Y|}$ to develop estimators $\widehat{TC}$ for triangle count TC, and to derive its concentration bounds.

As shown in Listing 1, TC can be obtained by summing intersections $|N_u \cap N_v|$ of neighborhoods for each pair of adjacent vertices u and v. Hence, to estimate TC, we simply sum cardinalities $\widehat{|N_u \cap N_v|}$ for each edge (u, v) in a given graph. This gives the following estimator:

$$\widehat{TC}_\star = \frac{1}{3} \sum_{(u,v)\in E} \widehat{|N_u \cap N_v|}_\star$$

where $\star$ indicates a specific $\widehat{|X \cap Y|}_\star$ estimator (cf. Table II).

Theorem VII.1. *Let $\widehat{TC}_\star$ be the estimator of the number of triangles. (cf. Section III). Then, depending on the underlying estimator $\widehat{|X \cap Y|}_\star$, we have the following cases:*

*For the **Bloom Filter** AND estimator, if $b\Delta \leq 0.499 B_X \log B_X$, then we have the following bound*

$$P\left(\left|TC - \widehat{TC}_{AND}\right| \geq t\right) \leq \frac{2\, m^2(1+o(1))\left(e^{\frac{\Delta b}{B_X - 1}}\frac{B_X}{b^2} - \frac{B_X}{b^2} - \frac{\Delta}{b}\right)}{9\, t^2}$$

*In the case of both **1-Hash** and k**-Hash** (below, we use the notation for 1-Hash), we have*

$$P\left(\left|TC - \widehat{TC}_{1H}\right| \geq t\right) \leq 2\exp\left(-\frac{18\, k\, t^2}{\left(\sum_{v\in V} d(v)^2\right)^2}\right)$$

Moreover, if the maximum degree is Δ, then

$$P\left(\left|TC - \widehat{TC}_{1H}\right| \geq t\right) \leq 2\exp\left(-\frac{9\, k\, t^2}{4\,(\Delta+1)\sum_{v\in V} d(v)^3}\right)$$

We provide a detailed proof of each statement of Theorem VII.1 in the Appendix.

Consistency of all the TC estimators follows from consistency of the individual estimators (cf. § II-F). The fact that $\widehat{TC}_{kH}$ is MLE follows from $\widehat{|X \cap Y|}_{kH}$ being MLE.

A. Comparison to Existing Estimators

We compare our $\widehat{TC}$ estimators to others in Table VII. We consider the estimators from Doulion [46], topological graph sketching [48], GAP [50], ASAP [49], Slim Graph [51], MCMC [108], and the "colorful" TC analysis [47], as well as several more recent results from the theory community [52], [53]. In the comparison, we consider construction time, used memory, TC estimation time, and whether an estimator is asymptotically unbiased, consistent, maximum likelihood, invariant, and whether it offers concentration bounds, and – if

[5] Link will be available upon publication due to double blindness.

	Reference	Constr. time	Memory used	Estimation time	AU	CN	ML	IN	AE	B
Past results	Doulion [46]	$O(m)$	$O(pm)$	$O(T(pm))$	👍	👍	✖	✖	✖	✖
	Colorful [47]	$O(m)$	$O(pm)$	$O(T(pm))$	👍	👍	✖	✖	✖	👍(P)
	Sketching [48]	$O(km)$	$O(kn)$	$O(T(k^2n))$	👍	👍	✖	✖	✖	✖
	ASAP [49]	n/a	$O(n+m)$	$O(1)$ / sample	✖	✖	✖	✖	✖	✖
	GAP [50]	$O(m)$†	$O(m')$†	$O(T(m'))$†	✖	✖	✖	✖	✖	✖
	Slim Gr. [51]	$O(m)$	$O(pm)$	$O(T(pm))$	👍	👍	✖	✖	✖	✖
	Eden et al. [52]	n/a	$O\left(\frac{n}{TC^{1/3}}\right)$	$O\left(\frac{n}{TC^{1/3}}+\frac{m^{3/2}}{TC}\right)$	👍	👍	✖	✖	✖	👍
	Assadi et al. [53]	n/a	$O(1)$	$O(m^{3/2}/TC)$	👍	👍	✖	✖	✖	👍
	Tětek [54]	n/a	$\left(\frac{m^{1.41}}{TC^{0.82}}\right)$	$\left(\frac{m^{1.41}}{TC^{0.82}}\right)$	👍	👍	✖	✖	✖	👍
PG	$\widehat{TC}_{AND}$ (BF)	$O(bm)$	$O(n+m)$	$O(mB/W)$	👍	👍	✖	✖	✖	👍(P)
	$\widehat{TC}_{kH}$ (MH)	$O(km)$	$O(n+m)$	$O(km)$	👍	👍	👍	👍	👍	👍(E)
	$\widehat{TC}_{1H}$ (MH)	$O(km)$	$O(n+m)$	$O(km)$	👍	👍	✖	✖	✖	👍(E)

TABLE VII: **ProbGraph vs. existing results for estimating TC** (sorted chronologically). **"Constr. time":** time to construct a given estimator. **"Memory":** the amount of storage needed to construct a given estimator. **"Estimation time":** time needed to estimate TC. **"AU" (asymptotically unbiased), "CN" (consistent)**, **"ML" (maximum likelihood estimator)**, **"IN" (invariant)**, **"AE" (asymptotically efficient)**: properties of estimators (explained in § II-F). **"B" (concentration bounds)**: whether a given scheme is supported with concentration bounds. **"P" (polynomial)** or **"E" (exponential)**: bound quality. "👍": supported, provided. "✖": not available, not provided. "†": the original work does *not* explicitly provide a given result and it was derived in this work. **Symbols used in related work (different from ones in Table I):** p: probability of keeping an edge, m': #sampled edges.

yes – are they polynomial or exponential (cf. Section II-F for an explanation on the relevance of these properties).

All ProbGraph's estimators offer polynomial or exponential concentration bounds. Importantly, $\widehat{TC}$ based on MinHash is the only one to offer exponential concentration bounds so far. This means that – for both $\widehat{TC}_{1H}$ and $\widehat{TC}_{kH}$ – any deviation from the true value of TC goes to zero exponentially fast with the increasing size of the potential deviation. Moreover, we observe that $\widehat{TC}_{kH}$ has all the desirable estimator properties mentioned above. Thus, it is particularly attractive whenever high accuracy is of the uttermost importance.

VIII. Evaluation

We now show that ProbGraph enables large speedups in graph mining while maintaining high accuracy of outcomes. We do not advocate a single way of deriving $|X \cap Y|$, but we illustrate pros and cons of different classes of schemes, and underline when each scheme is best applicable.

A. Datasets, Methodology, Architectures

We follow a recent set of recommendations on the benchmarking parallel applications [109]. For example, we omit the first 1% of performance data as warmup. We derive enough data for the mean and 95% non-parametric confidence intervals.

Comparison Baselines We compare PG-based approximate graph algorithms to tuned state-of-the-art implementations (Triangle and 4-Clique Counting, Clustering, and Vertex Similarity) from the GAP [105] and GMS [4] graph benchmarking suites. Moreover, when analyzing our estimators of $|N_u \cap N_v|$, we consider an existing BF estimator [110], [111], given by the expression $\widehat{|X|} = -\frac{\log(1-B_{X,1}/B_X)}{b\log(1-1/B_X)}$. We also consider another existing estimator [59], given by the expression $|\widehat{X \cap Y}|_{OR} = |X| + |Y| + \frac{B_{X\cup Y}}{b}\log\left(1 - \frac{B_{X\cup Y,1}}{B_{X\cup Y}}\right)$; this estimator uses the single set estimator evaluated on the set union. Finally, when evaluating TC, we compare to the established TC estimators: Doulion [46] (representing schemes based on edge sampling) colorful TC [47] (representing schemes based on sophisticated combinatorial pruning). We also consider state-of-the-art heuristics that do not come with theoretical guarantees: Reduced Execution [112], Partial Processing [112], Auto-Approximate (two variants) [113].

Datasets We consider SNAP (S) [114], KONECT (K) [115], DIMACS (D) [116], Network Repository (N) [117], and WebGraph (W) [118] datasets. For broad analysis, we follow the recommendations of the GMS graph mining benchmark [4], and we use networks of different origins (biology, chemistry, economy, etc.), sizes, densities (m/n), degree distribution skews, and even higher-order characteristics (e.g., counts of cliques). We illustrate the real-world datasets in Table VIII. We also use synthetic graphs power-law (the Kronecker model [119]) degree distribution. Using such synthetic graphs enables systematically changing a specific single graph property such as n, m, or m/n, which is not possible with real-world datasets. *This entails a very large evaluation space and we only include representative findings for selected graphs.*

Biological. Gene functional associations: (*bio-SC-GT*, 1.7K, 34K), (*bio-CE-PG*, 1.9K, 48K), (*bio-CE-GN*, 2.2K, 53.7K), (*bio-DM-CX*, 4K, 77K), (*bio-DR-CX*, 3.3K, 85K), (*bio-HS-LC*, 4.2K, 39K), (*bio-HS-CX*, 4.4K, 108.8K), (*bio-SC-HT*, 2K, 63K), (*bio-WormNetB3*, 2.4K, 79K). (*bio-WormNet-v3*, 16.3K, 762.8K). Human gene regulatory network: (*bio-humanGene*, 14K, 9M), (*bio-mouseGene*, 45K, 14.5M).
Interaction. Animal networks: (*int-antCol3-d1*, 161, 11.1K), (*int-antCol5-d1*, 153, 9K), (*int-antCol6-d2*, 165, 10.2K), (*intD-antCol4*, 134, 5K). Human contact network: (*int-HosWardProx*, 1.8k, 1.4k). Users-rate-users: (*int-dating*, 169K, 17.3M), (*edit-enwiktionary*, 2.1M, 5.5M). Collaboration: (*int-citAsPh*, 17.9K, 197K).
Brain. (*bn-flyMedulla*, 1.8K, 8.9K), (*bn-mouse*, 1.1K, 90.8K), (*bn-mouse_brain_1*, 213, 21.8K).
Economic. (*econ-psmigr1*, 3.1K, 543K), (*econ-psmigr2*, 3.1K, 540K), (*econ-beacxc*, 498, 50.4K), (*econ-beaflw*, 508, 53.4K), (*econ-mbeacxc*, 493, 49.9K), (*econ-orani678*, 2.5K, 90.1K).
Social. Facebook: (*soc-fbMsg*, 1.9k, 13.8k), Orkut: (3.1M, 117M).
Scientific computing. (*sc-pwtk*, 217.9K, 5.6M), (*sc-OptGupt*, 16.8K, 4.7M), (*sc-ThermAB*, 10.6K, 522.4K).
Discrete math. (*dimacs-c500-9*, 501, 112K), (*dimacs-hat1500-3*, 1.5K, 847K).
Chemistry. (*ch-SiO*, 33.4K, 675.5K), (*ch-Si10H16*, 17K, 446.5K).

TABLE VIII: Used graphs. For each graph, we show its "(#vertices, #edges)".

Parametrizing Set Representations We use the generic storage budget parameter s (cf. § V-A) to set the maximum allowable amount of memory than can be used by PG. Then, the parameters specific to each probabilistic set representation (b, B_X, k) enable fine tuning the tradeoff between storage, accuracy, and performance. In the following, we will also illustrate how to pick these parameters to maximize performance and accuracy for a given storage budget.

Architectures We use a a Dell PowerEdge R910 server with an Intel Xeon X7550 CPUs @ 2.00GHz with 18MB L3 cache, 1TiB RAM, and 32 cores per CPU (grouped in four sockets). We also use XC50 compute nodes in the Piz Daint Cray supercomputer (one such node comes with 12-core Intel Xeon E5-2690 HT-enabled CPU 64 GiB RAM).

Parallelism Unless stated otherwise, we run algorithms on the maximum number of cores available in a system.

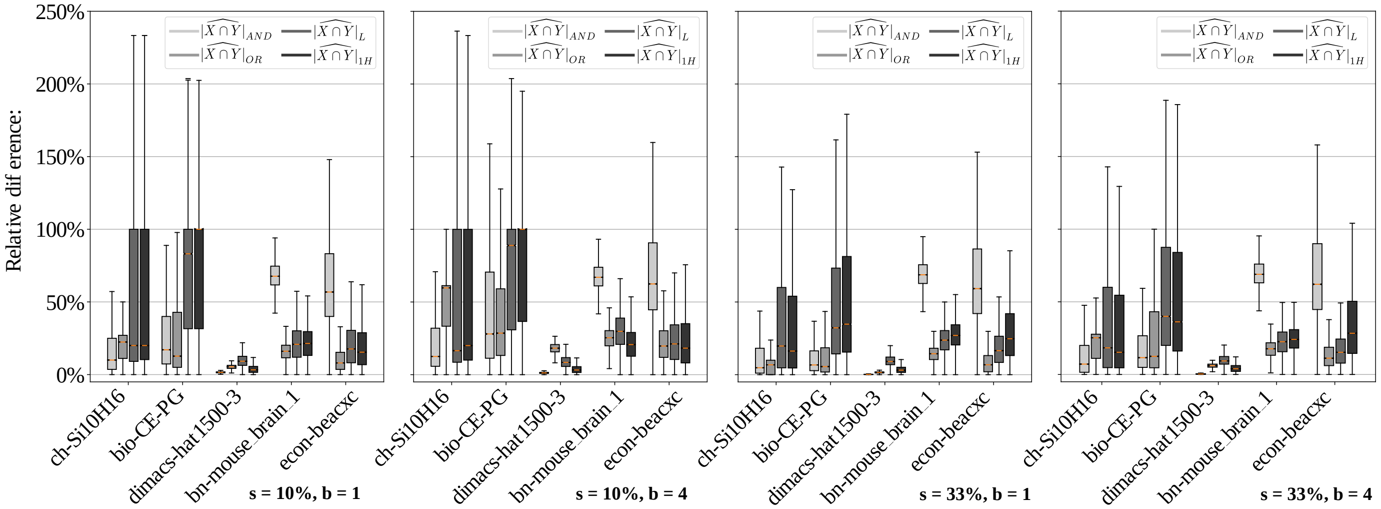


Fig. 3: Analysis of the accuracy of PG estimators of $|X \cap Y|$.

Assessing Accuracy To measure the accuracy of algorithms that return some *counts* (e.g., clique count, count of clusters), we use expression $\frac{|cnt_{PG}-cnt_{EX}|}{cnt_{EX}}$, where cnt_{PG} and cnt_{EX} are ProbGraph and exact counts, respectively. Note that the ProbGraph counts may be lower but also higher than the exact ones (due to false positives in BFs).

B. Estimating $|N_v \cap N_u|$

We first assess specific PG estimators of $|N_v \cap N_u|$ in terms of their accuracy. For each graph, we derive the BF and MH representations of its vertex neighborhoods, and then the intersections of neighborhoods of adjacent vertices. Finally, we compute the relative differences between these PG intersection cardinalities and the CSR related cardinalities $|(\widehat{|X \cap Y|}_{\bullet} - |X \cap Y|)|/|X \cap Y|$ where $\bullet \in \{AND, L, 1H, kH\}$ We summarize these differences, for each graph, using boxplots. We use the storage budget $s = 33\%$ and $b \in \{1, 4\}$.

Representative results are in Figure 3. While medians are low (less than ≈25% for most cases), there is a certain spread in outliers. This is because we consider *all* adjacent vertices, and there is a high chance that at least some pairs will result in low accuracy. Overall, the results illustrate that there is no single winner among the estimators, and the outcomes depend on the graph structure. One observation is that the BF based on AND tends to perform worse on very dense graphs, and comparably to marginally better than L on sparser graphs. Similarly, k–Hash is marginally worse than 1–Hash on very dense graphs; sparse graphs entail a reverse pattern.

C. Estimating Outcomes of Graph Algorithms

We now illustrate that PG estimators enable high performance and high accuracy at a small additional storage budget when applied to parallel graph mining. We first conduct an analysis using all 32 cores. For each graph problem considered, we illustrate the exact baseline and the schemes based on BF and MH estimators. We use $\widehat{|N_u \cap N_v|}_{AND}$ with $b = 2$ and $\widehat{|N_u \cap N_v|}_{1H}$ that represent BF and MH schemes; they offer high accuracy while being fast to compute as they need few hash functions (other estimators come with similar accuracy outcomes but are slower to compute). Figures 4 and 5 show the results for real-world and Kronecker graphs. Each single plot is dedicated to a specific graph problem and it compares different estimators (indicated with different shapes of data points) across three dimensions: performance (speedup, X axis), accuracy (relative count, Y axis), and memory budget (relative memory size with respect to the default CSR, shades of B&W). Each plot corresponds to a specific graph problem. Each data point corresponds to the execution of a given scheme for a specific graph dataset. Thus, each plot shows collectively how different baselines behave for different input graphs.

In general, the results follow the insights from the analysis of estimating set intersections. BFs offer high accuracy and high speedups, sometimes even as high as 20× (Clustering using Overlap), or nearly 30× (Clustering using Common Neighbors), while keeping the accuracy more than 98%. Speedups can be as high as 50×, with the accuracy more than 90% (e.g., 4-Clique Counting for Kronecker graphs). On the other hand, MH usually gives consistently higher speedups as well as lower memory requirements, *but its accuracy is in most cases worse than BF*. We conjecture this is because MH estimators preserve only specific subsets of vertex neighborhoods (selected using hash functions), explicitly eliminating other vertices. In contrast, when using BF estimators, each vertex is hashed to certain bit(s) in the final bit vector, and is thus to some extent "reflected" in PG estimators.

In terms of memory efficiency, the mostly very light shades indicate very low additional storage overheads. Except for a few cases where light gray indicates that – for a given graph – a given estimator needs around 20-50% additional space, *all the cases require at most 25% more storage*. We additionally indicate this in the plots with the appropriate annotations.

We then provide more details on all the problems using

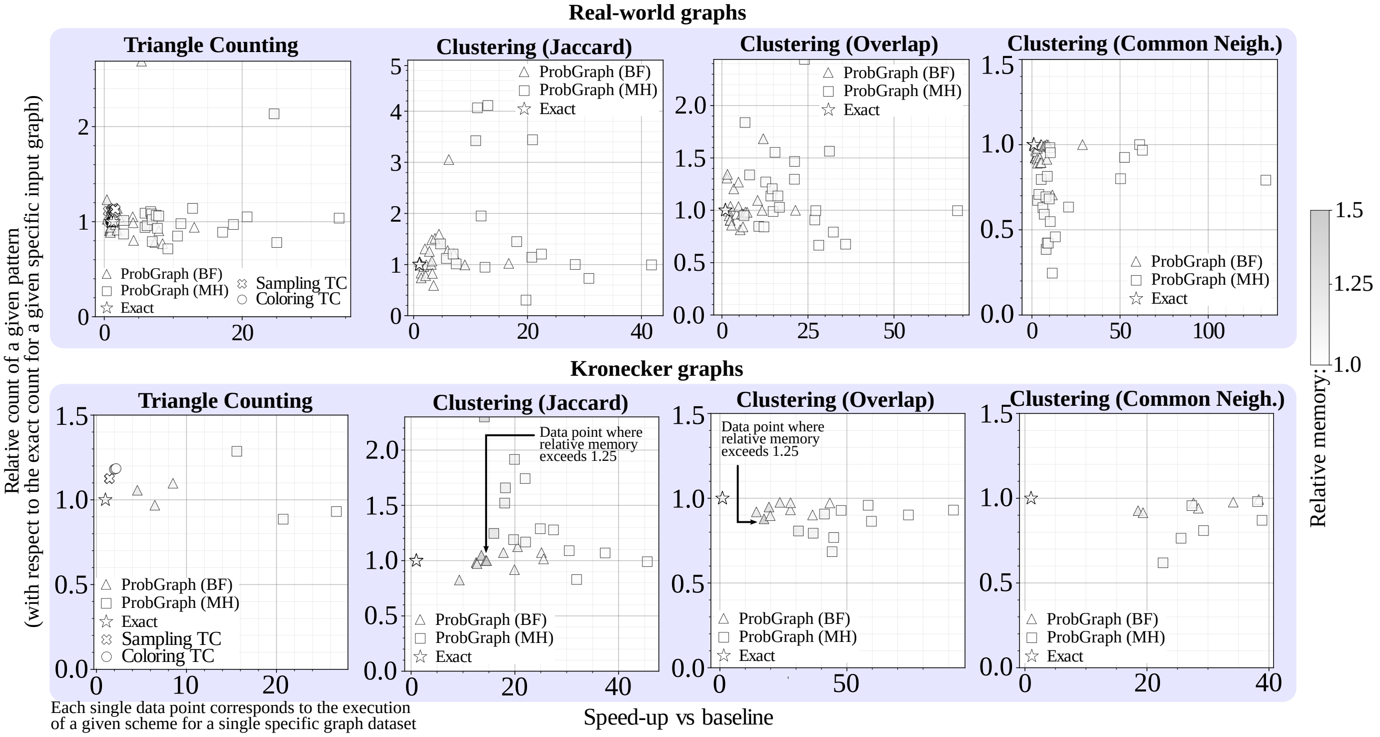


Fig. 4: Summary of advantages of PG for real-world (top panel) and Kronecker (bottom panel) graphs, for Triangle Counting and Clustering. All 32 cores are used. Note that most data points are white or almost white because they come with very low amounts of additional memory (we annotate a few data points that come with more than 25% additional relative memory amounts). Other graph problems come with the same performance/memory/accuracy patterns for the used comparison baselines.

detailed bar plots; due to space constraints, we only show TC in Figure 6 for real-world graphs (all other problems and Kronecker graphs come with similar insights). For the storage budget of at most 25%, for the majority of graphs – regardless of their origin – PG enables high (>80%) accuracy combined with high speedups. The only cases of low accuracy (i.e., the number of clusters detected being much higher than the exact one) is when using MH. This is because MH based schemes explicitly remove a (usually high) number of edges, resulting in possibly significant increase in cluster counts.

We conclude that BF-based PG estimators consistently deliver high accuracy as well as speedups at small memory budget, for a broad set of graphs and problems. 1–Hash based schemes may provide much more performance, but require more careful parametrization and input selection.

D. Comparison to Heuristics

We also compare to heuristics for approximate graph computations that do not come with guarantees on the quality of outcomes. One scheme called “Reduced Execution” [112] reduces the count of iterations of the outermost loop. Another scheme, “Partial Graph Processing” [112], processes – for each vertex v – a randomly selected subset of v’s neighbors. Moreover, we use two variants of sampling-based “Auto-Approximation” that addresses a purely vertex-centric model of computation [113]. We show these heuristics in Figure 6 (we exclude them from Figures 4 and 5 to preserve clarity; they always achieve much worse results that obscure plots). The advantage of heuristics is that they do not need additional memory, as shown in the lowest panel in Figure 6. However, PG always achieves much better accuracy, by at least 25%, up to ≈75%. This is because the heuristics are not based on theoretical developments that ensure high PG’s accuracy. Moreover, the heuristics are also slower than PG. “AutoApproximate” schemes introduce particularly large overheads due to their purely vertex-centric abstraction, which makes them even slower than the exact tuned baselines that we compare

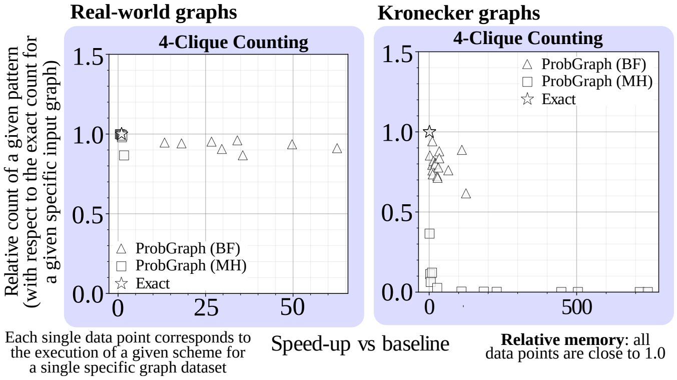


Fig. 5: Summary of advantages of PG for 4-clique counting for real-world (left panel) and Kronecker (right panel) graphs. All 32 cores are used.

to. The results for all other graph problems are similar.

E. Analysis of Scaling

We also consider strong/weak scaling; Figure 8 offers representative results. We observe nearly ideal strong scaling of all the baselines compared. PG schemes feature much lower runtimes. To investigate in what regime of parameters PG baselines exhibit better scaling behavior, we also analyze weak scaling, see Figure 9b. We use Kronecker graphs. We increase the number of edges along with the number of threads, from $m \approx$ 4M to $m \approx$ 1.8B for a fixed $n =$ 1M. The largest graphs fill the whole available memory (1TB). In this experiment, we increase the number of edges at a *rate twice as large as the thread count* (cf. the X axis). As we use Kronecker graphs, this *stresses load balancing capabilities of the compared baselines*, as most vertices have small neighborhoods, but some neighborhoods grow particularly fast, making it very challenging to load balance set intersections (cf. the right side of Figure 1). The results illustrate that all PG baselines scale *much better than* all competition baselines. It becomes particularly visible beyond a certain point, where the PG scaling curves become gradually flatter. This is enabled by the PG design, in which set representations are of the same (usually very small) size. Hence, load imbalance is less of an issue, while – as shown earlier in this section – accuracy loss is negligible. Finally, Figure 9 shows that the difference between BF and MH in scaling also depends on the targeted problem. For Clustering based on Common Neighbors, BF becomes comparable, or marginally better, than MH, for large thread counts. This is because the algorithm for Clustering is almost completely dominated by $|X \cap Y|$, hence benefiting from BF's very fast bitwise AND set intersections.

F. Analysis of Construction Costs

We also analyze the *construction costs* of PG. Time to construct a single neighborhood follows asymptotic complexities in Table V; it is not a bottleneck and is lower than 50% of the algorithm execution time in the majority of cases. Only using very large b may bring the preprocessing time larger than the single graph algorithm execution time, but (1) PG benefits from low $b \in \{1, 2\}$ in any case, and (2) the PG representation of a graph has to be computed only once, and it can be then freely used with any considered graph algorithms.

IX. Beyond Bloom Filter and MinHash

The generic nature of PG enables using other probabilistic representations in place of BF and MH. As an example, we discuss how to use PG with *K Minimum Values* (KMV) [120], another sketch that was originally developed to accelerate counting distinct elements in a data stream. To construct a KMV representation $\mathcal{K}_X$ of a set X, one evaluates the associated hash function $h : X \to (0; 1]$ for all elements of X. Then, one selects k *smallest* hashes that constitute the final KMV representation $\mathcal{K}_X$ of the set X. One can then estimate $|X|$ with $\widehat{|X|}_{KMV} = \frac{k-1}{\max \mathcal{K}_X}$. Note that $\mathcal{K}_X$ differs from a MH $\mathcal{M}_X$ because, as opposed to $\mathcal{M}_X$, it contains hashes.

Now, one can use KMV to also estimate $|X \cap Y|$, and then use it within PG. For this, one constructs a KMV $\mathcal{K}_{X \cup Y}$ by taking the k smallest elements from $K_X \cup K_Y$. Then, by the KMV properties, we have $\widehat{|X \cup Y|}_{KMV} = \frac{k-1}{\max \mathcal{K}_{X \cup Y}}$. Finally, $\widehat{|X \cap Y|}_{KMV} = |X| + |Y| - \widehat{|X \cup Y|}_{KMV}$, which can be directly used in PG formulations of graph algorithms. We provide concentration bounds for all the KMV estimators defined above in the Appendix.

X. Related Work: Summary

We summarize related work; some parts are already covered in Sections I and VII. First, there exist more set-related probabilistic data structures, for example HyperLogLog [121]. *ProbGraph embraces such data structures:* while we focus on BF [35] and MH [37], one could easily extend ProbGraph with other structures; we leave details for future work.

Next, there are many approximate graph algorithms [20]–[24], [26]–[30], [30]–[33], [49]. ProbGraph differs from them as it can approximate any algorithm or scheme that uses $|X \cap Y|$, set membership query, and others, where X and Y are arbitrary sets of vertices or edges (all our theoretical and most of empirical results are directly applicable to any sets). Moreover, ProbGraph is simple: all one has to do is to plug in a selected set representation.

A few existing general approaches for approximate graph computations usually target specific problems or they do not come with guarantees on the quality of outcomes [108], [112], [113]. As shown in Section VII, unlike ProbGraph, specific schemes for TC do not offer strong accuracy guarantees [46]–[51], [108].

ProbGraph's probabilistic representations of vertex sets are a form of graph compression [122], and they could be used to extend existing compressed graph representations and paradigms [51], [123].

There exist a few works on using BF or MH specific single graph problems, usually in the context of evolving graphs [48], [124]–[127], which is outside PG's scope.

Approximating the triangle count in time less then linear in the size of the input was shown in [52]. This has been later generalized to approximating the number of k-cliques in a graph [128]. Improved bounds are known when the arboricity of the graph is small [129]. Assuming we can sample edges uniformly, better algorithms are also known [53], [130]. Unlike PG, these schemes are specific to selected graph problems and graphs with certain properties such as low arboricity.

There are many works on counting or finding different graph patterns (also called motifs, graphlets, or subgraphs) [1], [3], [4], [9], [12], [14], [68], [74], [75], [97], [131]–[144]. PG can be used as a subroutine in different such works, offering speedups while preserving high accuracy.

Counting and listing simple patterns such as triangles have been recently used to enhance the design of numerous models in Graph Neural Networks [145], [145], [146], [146]–[148], [148], [149], [149], [150], [150], [151], [151]–[156]. Such models could use PG to accelerate expensive graph mining preprocessing costs.

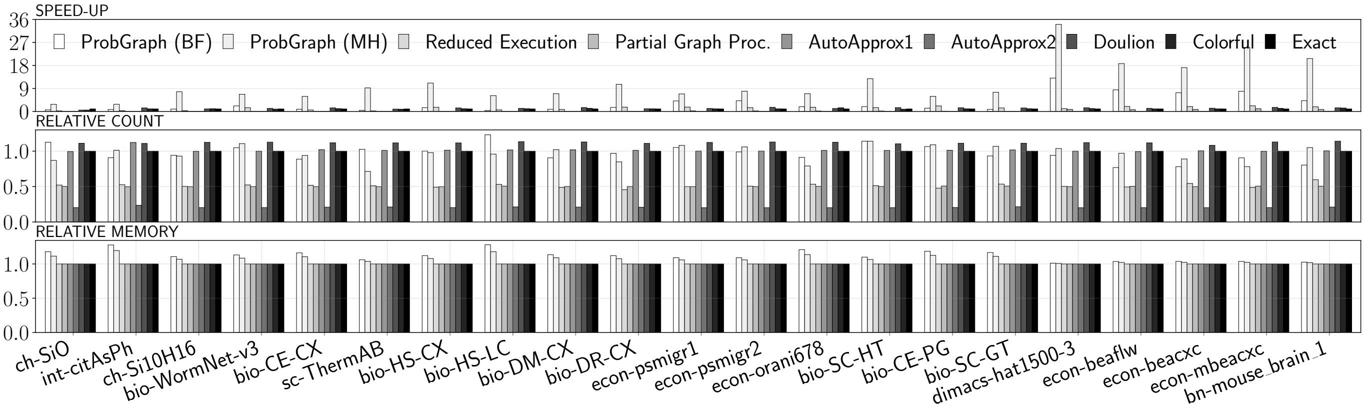


Fig. 6: Analysis of performance/accuracy/memory of ProbGraph for Triangle Counting, illustrating advantages of ProbGraph over baselines with theoretical underpinning (Sampling, Colorful) and over heuristics (Reduced Execution, Partial Graph Processing, AutoApprox1, AutoApprox2).

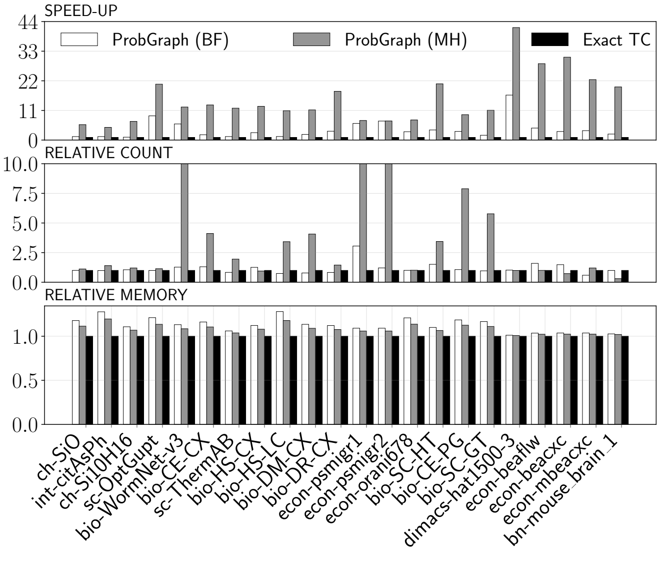


Fig. 7: Analysis of performance/accuracy/memory of ProbGraph, for Clustering based on the Jaccard Coefficient score for Vertex Similarity. For relative counts of clusters, we set a cutoff for the value of 10 for clarity of plots.

The straightforward parallelism in computing BF based estimators implies that other architectures that offer massive parallelism may provide even higher benefits. This includes FPGAs [27], [157]–[159], CGRAs [160], or processing in-memory [161]–[169]. We leave these studies for future work.

Next, there are many **approximate graph algorithms** [18], [22], [170], [170]–[173]. ProbGraph differs from them as *it is general*: it can approximate any algorithm or scheme that uses $|X \cap Y|$, set membership query, and others, where X and Y are arbitrary sets of vertices or edges (all our theoretical and most of empirical results are directly applicable to any sets). Moreover, ProbGraph *is simple*: all one has to do is to plug in a selected set representation and implementations of $|X \cap Y|$, a set membership query, and any other related schemes.

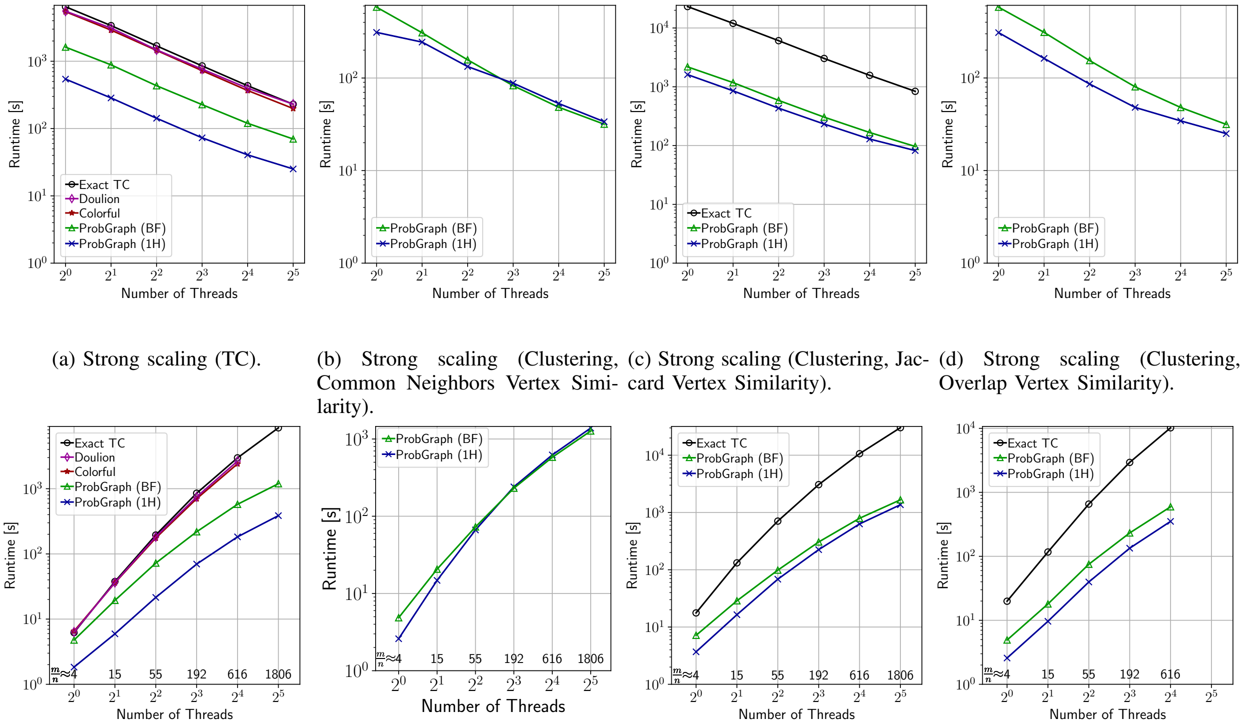
XI. Conclusion

We propose ProbGraph, a parallel graph representation that enables simple, general, and high-performance approximate graph computations. The key idea is to sketch sets of vertices, and the cardinality of the intersection of such sets, with probabilistic set representations such as Bloom filters or MinHash. Such representations usually offer much higher performance than exact set representations, while only requiring small additional storage. Importantly, they can be treated as a black box and seamlessly incorporated into graph algorithms. We show that ProbGraph is simple to use while offering speedups of more than 50× for some graphs and retaining high accuracy of more than 90% for problems such as Triangle Counting, when comparing to tuned exact parallel baselines on 32 cores.

We support ProbGraph with an in-depth theoretical underpinning, in which we derive novel statistical concentration bounds on the accuracy of ProbGraph approximations. Our bounds are the first exponential or polynomial quality bounds for the accuracy of Bloom filters and MinHash. As such, they are of interest to the broader audience beyond graph analytics. We also use the work-depth formal analysis to show that ProbGraph has also theoretical advantages over parallel baselines in parallel computational complexity.

Set algebra is common in many graph problems. Hence, we expect that ProbGraph and its set-centric approach for approximate graph analytics may be used for other problems.

Acknowledgements: We thank Hussein Harake, Colin McMurtrie, Mark Klein, Angelo Mangili, and the whole CSCS team granting access to the Ault and Daint machines, and for their excellent technical support. We thank Timo Schneider for immense help with computing infrastructure at SPCL. We thank Aryaman Fasciati and Sebastian Leisinger for help in the early stages of the project. This research received funding from the European Research Council (Project DAPP, No. 678880; Project PSAP, No. 101002047), Huawei, and MS Research through its Swiss Joint Research Centre.



(a) Strong scaling (TC).

(b) Strong scaling (Clustering, Common Neighbors Vertex Similarity).

(c) Strong scaling (Clustering, Jaccard Vertex Similarity).

(d) Strong scaling (Clustering, Overlap Vertex Similarity).

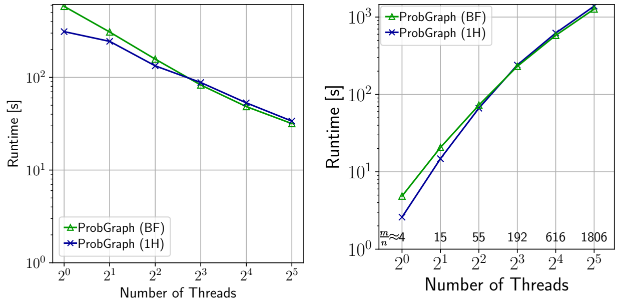
(e) Weak scaling (TC).

(f) Weak scaling (Clustering, Common Neighbors Vertex Similarity).

(g) Weak scaling (Clustering, Jaccard Vertex Similarity).

(h) Weak scaling (Clustering, Overlap Vertex Similarity).

Fig. 8: Analysis of scaling of representative baselines.



(a) Strong scaling.

(b) Weak scaling.

Fig. 9: Scaling results for Clustering (Common Neighbors), illustrating comparable scaling performance of both BF and MH.

Jakub Tětek was supported by the VILLUM Foundation grant 16582.

References

[1] M. Besta *et al.*, "Sisa: Set-centric instruction set architecture for graph mining on processing-in-memory systems," *arXiv preprint arXiv:2104.07582*, 2021.

[2] S. Arora, "A survey on graph neural networks for knowledge graph completion," *arXiv preprint arXiv:2007.12374*, 2020.

[3] M. Besta, R. Grob, C. Miglioli, N. Bernold, G. Kwasniewski, G. Gjini, R. Kanakagiri, S. Ashkboos, L. Gianinazzi, N. Dryden *et al.*, "Motif prediction with graph neural networks," in *ACM KDD*, 2022.

[4] M. Besta *et al.*, "Graphminesuite: Enabling high-performance and programmable graph mining algorithms with set algebra," *arXiv preprint arXiv:2103.03653*, 2021.

[5] A. Lumsdaine, D. Gregor, B. Hendrickson, and J. W. Berry, "Challenges in Parallel Graph Processing," *Par. Proc. Let.*, vol. 17, no. 1, pp. 5–20, 2007.

[6] S. Sakr *et al.*, "The future is big graphs! a community view on graph processing systems," *arXiv preprint arXiv:2012.06171*, 2020.

[7] M. Besta, "Enabling high-performance large-scale irregular computations," Ph.D. dissertation, ETH Zurich, 2021.

[8] A. Tate *et al.*, "Programming abstractions for data locality," in *PADAL Workshop*. PADAL Workshop 2014, 2014.

[9] M. Besta, M. Podstawski, L. Groner, E. Solomonik, and T. Hoefler, "To push or to pull: On reducing communication and synchronization in graph computations," in *ACM HPDC*. ACM, 2017, pp. 93–104. [Online]. Available: https://doi.org/10.1145/3078597.3078616

[10] M. Besta and T. Hoefler, "Accelerating irregular computations with hardware transactional memory and active messages," in *ACM HPDC*, 2015, pp. 161–172. [Online]. Available: https://doi.org/10.1145/2749246.2749263

[11] D. J. Cook and L. B. Holder, *Mining graph data*. John Wiley & Sons, 2006.

[12] C. Jiang, F. Coenen, and M. Zito, "A survey of frequent subgraph mining algorithms," *The Knowledge Engineering Review*, vol. 28, no. 1, pp. 75–105, 2013.

[13] T. Horváth, T. Gärtner, and S. Wrobel, "Cyclic pattern kernels for predictive graph mining," in *KDD*. ACM, 2004, pp. 158–167.

[14] D. Chakrabarti and C. Faloutsos, "Graph mining: Laws, generators, and algorithms," *ACM computing surveys (CSUR)*, vol. 38, no. 1, p. 2, 2006.

[15] S. Mittal, "A survey of recent prefetching techniques for processor caches," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, p. 35, 2016.

[16] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design & Test*, vol. 33, no. 1, pp. 8–22, 2015.

[17] J. Han and M. Orshansky, "Approximate computing: An emerging

paradigm for energy-efficient design," in *Test Symposium (ETS), 2013 18th IEEE European*. IEEE, 2013, pp. 1–6.

[18] M. M. Halldórsson, "A still better performance guarantee for approximate graph coloring," *Information Processing Letters*, vol. 45, no. 1, pp. 19–23, 1993.

[19] M. T. Jones and P. E. Plassmann, "A parallel graph coloring heuristic," *SIAM Journal on Scientific Computing*, vol. 14, no. 3, pp. 654–669, 1993.

[20] M. Riondato and E. M. Kornaropoulos, "Fast approximation of betweenness centrality through sampling," *Data Mining and Knowledge Discovery*, vol. 30, no. 2, pp. 438–475, 2016.

[21] M. Borassi and E. Natale, "Kadabra is an adaptive algorithm for betweenness via random approximation," *arXiv preprint arXiv:1604.08553*, 2016.

[22] M. Riondato and E. Upfal, "Abra: Approximating betweenness centrality in static and dynamic graphs with rademacher averages," *ACM TKDD*, vol. 12, no. 5, p. 61, 2018.

[23] R. Geisberger, P. Sanders, and D. Schultes, "Better approximation of betweenness centrality," in *Proceedings of the Meeting on Algorithm Engineering & Expermiments*. Society for Industrial and Applied Mathematics, 2008, pp. 90–100.

[24] D. A. Bader *et al.*, "Approximating betweenness centrality," in *Algorithms and Models for the Web-Graph*. Springer, 2007, pp. 124–137.

[25] E. Solomonik, M. Besta, F. Vella, and T. Hoefler, "Scaling betweenness centrality using communication-efficient sparse matrix multiplication," in *ACM/IEEE Supercomputing*. ACM, 2017, p. 47. [Online]. Available: https://doi.org/10.1145/3126908.3126971

[26] B. Chazelle, R. Rubinfeld, and L. Trevisan, "Approximating the minimum spanning tree weight in sublinear time," *SIAM Journal on computing*, vol. 34, no. 6, pp. 1370–1379, 2005.

[27] M. Besta, M. Fischer, T. Ben-Nun, D. Stanojevic, J. D. F. Licht, and T. Hoefler, "Substream-centric maximum matchings on fpga," *ACM TRETS*, vol. 13, no. 2, pp. 1–33, 2020.

[28] S. Dumbrava, A. Bonifati, A. N. R. Diaz, and R. Vuillemot, "Approximate evaluation of label-constrained reachability queries," *arXiv preprint arXiv:1811.11561*, 2018.

[29] S. Chechik, D. H. Larkin, L. Roditty, G. Schoenebeck, R. E. Tarjan, and V. V. Williams, "Better approximation algorithms for the graph diameter," in *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2014, pp. 1041–1052.

[30] L. Roditty and V. Vassilevska Williams, "Fast approximation algorithms for the diameter and radius of sparse graphs," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 2013, pp. 515–524.

[31] G. M. Slota and K. Madduri, "Complex network analysis using parallel approximate motif counting," in *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*. IEEE, 2014, pp. 405–414.

[32] P. Boldi, M. Rosa, and S. Vigna, "Hyperanf: Approximating the neighbourhood function of very large graphs on a budget," in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 625–634.

[33] G. ECHBARTHI and H. KHEDDOUCI, "Lasas: an aggregated search based graph matching approach," in *The 29th International Conference on Software Engineering and Knowledge Engineering*, 2017.

[34] M. Besta, A. Carigiet, K. Janda, Z. Vonarburg-Shmaria, L. Gianinazzi, and T. Hoefler, "High-performance parallel graph coloring with strong guarantees on work, depth, and quality," in *ACM/IEEE Supercomputing*, 2020, pp. 1–17.

[35] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *CACM*, vol. 13, no. 7, pp. 422–426, 1970.

[36] S. Han, L. Zou, and J. X. Yu, "Speeding up set intersections in graph algorithms using simd instructions," in *Proceedings of the 2018 International Conference on Management of Data*. ACM, 2018, pp. 1587–1602.

[37] A. Z. Broder, "On the resemblance and containment of documents," in *SEQUENCES*. IEEE, 1997.

[38] S. Boucheron, G. Lugosi, and P. Massart, *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

[39] G. Casella and R. L. Berger, *Statistical inference*. Duxbury Pacific Grove, CA, 2002, vol. 2.

[40] V. Estivill-Castro, "Why so many clustering algorithms: a position paper," *ACM SIGKDD explorations newsletter*, vol. 4, no. 1, pp. 65–75, 2002.

[41] C. C. Aggarwal and H. Wang, "A survey of clustering algorithms for graph data," in *Managing and mining graph data*. Springer, 2010, pp. 275–301.

[42] S. E. Schaeffer, "Graph clustering," *Computer science review*, vol. 1, no. 1, pp. 27–64, 2007. [Online]. Available: https://doi.org/10.1016/j.cosrev.2007.05.001

[43] L. Gianinazzi, M. Besta, Y. Schaffner, and T. Hoefler, "Parallel algorithms for finding large cliques in sparse graphs," 2021.

[44] R. A. Jarvis and E. A. Patrick, "Clustering using a similarity measure based on shared near neighbors," *IEEE Transactions on computers*, vol. 100, no. 11, pp. 1025–1034, 1973.

[45] M. Besta *et al.*, "Communication-efficient jaccard similarity for high-performance distributed genome comparisons," in *IEEE IPDPS*. IEEE, 2020, pp. 1122–1132.

[46] C. E. Tsourakakis *et al.*, "Doulion: counting triangles in massive graphs with a coin," in *ACM KDD*, 2009.

[47] R. Pagh and C. E. Tsourakakis, "Colorful triangle counting and a mapreduce implementation," *Information Processing Letters*, vol. 112, no. 7, pp. 277–281, 2012.

[48] B. Bandyopadhyay *et al.*, "Topological graph sketching for incremental and scalable analytics," in *CIKM*, 2016, pp. 1231–1240.

[49] A. P. Iyer, Z. Liu, X. Jin, S. Venkataraman, V. Braverman, and I. Stoica, "{ASAP}: Fast, approximate graph pattern mining at scale," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 745–761.

[50] A. P. Iyer *et al.*, "Bridging the gap: towards approximate graph analytics," in *ACM GRADES-NDA*, 2018.

[51] M. Besta *et al.*, "Slim graph: Practical lossy graph compression for approximate graph processing, storage, and analytics," pp. 1–25, 2019. [Online]. Available: https://doi.org/10.1145/3295500.3356182

[52] T. Eden, A. Levi, D. Ron, and C. Seshadhri, "Approximately counting triangles in sublinear time," *SIAM Journal on Computing*, vol. 46, no. 5, pp. 1603–1646, 2017.

[53] S. Assadi, M. Kapralov, and S. Khanna, "A simple sublinear-time algorithm for counting arbitrary subgraphs via edge sampling," *arXiv preprint arXiv:1811.07780*, 2018.

[54] J. Tětek, "Approximate triangle counting via sampling and fast matrix multiplication," *arXiv preprint arXiv:2104.08501*, 2021.

[55] G. Bilardi and A. Pietracaprina, *Models of Computation, Theoretical*. Boston, MA: Springer US, 2011, pp. 1150–1158.

[56] G. E. Blelloch and B. M. Maggs, *Parallel Algorithms*, 2nd ed. Chapman & Hall/CRC, 2010, p. 25.

[57] P. Bose *et al.*, "On the false-positive rate of bloom filters," *Information Processing Letters*, 2004.

[58] A. Z. Broder, "Identifying and filtering near-duplicate documents," in *CPM*, 2000.

[59] S. J. Swamidass and P. Baldi, "Mathematical correction for fingerprint similarity measures to improve chemical retrieval," *Journal of chemical information and modeling*, vol. 47, no. 3, pp. 952–964, 2007.

[60] I. J. Myung, "Tutorial on maximum likelihood estimation," *Journal of mathematical Psychology*, vol. 47, no. 1, pp. 90–100, 2003.

[61] R. A. Fisher, "On the mathematical foundations of theoretical statistics," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 222, no. 594-604, pp. 309–368, 1922.

[62] H. White, "Maximum likelihood estimation of misspecified models," *Econometrica: Journal of the econometric society*, pp. 1–25, 1982.

[63] V. Bentkus *et al.*, "On hoeffding's inequalities," *The Annals of Probability*, vol. 32, no. 2, pp. 1650–1673, 2004.

[64] E. Greene and J. A. Wellner, "Exponential bounds for the hypergeometric distribution," *Bernoulli: official journal of the Bernoulli Society for Mathematical Statistics and Probability*, vol. 23, no. 3, p. 1911, 2017.

[65] J. Shun and K. Tangwongsan, "Multicore triangle computations without tuning," in *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*. IEEE, 2015, pp. 149–160.

[66] A. Strausz, F. Vella, S. Di Girolamo, M. Besta, and T. Hoefler, "Asynchronous distributed-memory triangle counting and lcc with rma caching," 2022.

[67] M. Al Hasan and V. S. Dave, "Triangle counting in large networks: a review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 2, p. e1226, 2018.

[68] M. Danisch, O. Balalau, and M. Sozio, "Listing k-cliques in sparse real-world graphs," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2018, pp. 589–598.

[69] I. Robinson, J. Webber, and E. Eifrem, *Graph databases*. " O'Reilly Media, Inc.", 2013.

[70] ——, *Graph databases: new opportunities for connected data*. " O'Reilly Media, Inc.", 2015.

[71] M. Lissandrini, M. Brugnara, and Y. Velegrakis, "An evaluation methodology and experimental comparison of graph databases," Technical report, University of Trento, Tech. Rep., 2017.

[72] I. Robinson, J. Webber, and E. Eifrem, "Graph database internals," in *Graph Databases, Second Edition*. O'Relly, 2015, ch. 7, pp. 149–170.

[73] M. Besta, E. Peter, R. Gerstenberger, M. Fischer, M. Podstawski, C. Barthels, G. Alonso, and T. Hoefler, "Demystifying graph databases: Analysis and taxonomy of data organization, system designs, and graph queries," *arXiv preprint arXiv:1910.09017*, 2019.

[74] E. A. Leicht, P. Holme, and M. E. Newman, "Vertex similarity in networks," *Physical Review E*, vol. 73, no. 2, p. 026120, 2006.

[75] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[76] L. Wang, K. Hu, and Y. Tang, "Robustness of link-prediction algorithm based on similarity and application to biological networks," *Current Bioinformatics*, vol. 9, no. 3, pp. 246–252, 2014.

[77] A. Friggeri, G. Chelius, and E. Fleury, "Triangles to capture social cohesion," in *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*. IEEE, 2011, pp. 258–265.

[78] A. Prat-Pérez, D. Dominguez-Sal, J. M. Brunat, and J.-L. Larriba-Pey, "Shaping communities out of triangles," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 1677–1681.

[79] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis, "Efficient semi-streaming algorithms for local triangle counting in massive graphs," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 16–24.

[80] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[81] Z. Bar-Yossef, R. Kumar, and D. Sivakumar, "Reductions in streaming algorithms, with an application to counting triangles in graphs," in *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2002, pp. 623–632.

[82] J.-P. Eckmann and E. Moses, "Curvature of co-links uncovers hidden thematic layers in the world wide web," *Proceedings of the national academy of sciences*, vol. 99, no. 9, pp. 5825–5829, 2002.

[83] N. H. Tran, K. P. Choi, and L. Zhang, "Counting motifs in the human interactome," *Nature communications*, vol. 4, no. 1, pp. 1–8, 2013.

[84] K. Sugiyama, K. Hatano, and M. Yoshikawa, "Adaptive web search based on user profile constructed without any effort from users," in *Proceedings of the 13th international conference on World Wide Web*, 2004, pp. 675–684.

[85] N. A. Kratochwil, W. Huber, F. Müller, M. Kansy, and P. R. Gerber, "Predicting plasma protein binding of drugs: a new approach," *Biochemical pharmacology*, vol. 64, no. 9, pp. 1355–1374, 2002.

[86] P. Willett, J. M. Barnard, and G. M. Downs, "Chemical similarity searching," *Journal of chemical information and computer sciences*, vol. 38, no. 6, pp. 983–996, 1998.

[87] P. V. Kharchenko, "The triumphs and limitations of computational methods for scrna-seq," *Nature Methods*, vol. 18, no. 7, pp. 723–732, 2021.

[88] R. C. Tyser, E. Mahammadov, S. Nakanoh, L. Vallier, A. Scialdone, and S. Srinivas, "Single-cell transcriptomic characterization of a gastrulating human embryo," *Nature*, vol. 600, no. 7888, pp. 285–289, 2021.

[89] R. A. Hanneman and M. Riddle, "Introduction to social network methods," 2005.

[90] M. O. Jackson, "Social and economic networks," in *Social and Economic Networks*. Princeton university press, 2010.

[91] Z. Lu, J. Wahlström, and A. Nehorai, "Community detection in complex networks via clique conductance," *Scientific reports*, vol. 8, no. 1, pp. 1–16, 2018.

[92] M. Mitzenmacher, J. Pachocki, R. Peng, C. Tsourakakis, and S. C. Xu, "Scalable large near-clique detection in large-scale networks via sampling," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 815–824.

[93] A. E. Sariyuce, C. Seshadhri, A. Pinar, and U. V. Catalyurek, "Finding the hierarchy of dense subgraphs using nucleus decompositions," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 927–937.

[94] C. Tsourakakis, "The k-clique densest subgraph problem," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1122–1132.

[95] C. E. Tsourakakis, J. Pachocki, and M. Mitzenmacher, "Scalable motif-aware graph clustering," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 1451–1460.

[96] A. Sizemore, C. Giusti, and D. S. Bassett, "Classification of weighted networks through mesoscale homological features," *Journal of Complex Networks*, vol. 5, no. 2, pp. 245–273, 2017.

[97] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[98] M. Al Hasan *et al.*, "Link prediction using supervised learning," in *SDM06: workshop on link analysis, counter-terrorism and security*, 2006.

[99] B. Taskar, M.-F. Wong, P. Abbeel, and D. Koller, "Link prediction in relational data," in *Advances in neural information processing systems*, 2004, pp. 659–666.

[100] O. Ertl, "Bagminhash-minwise hashing algorithm for weighted sets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1368–1377.

[101] A. Z. Broder *et al.*, "Min-wise independent permutations," *J. Comp. and Sys. Sciences*, 2000.

[102] M. Besta, F. Marending, E. Solomonik, and T. Hoefler, "Slimsell: A vectorizable graph representation for breadth-first search," in *IEEE IPDPS*. IEEE, 2017, pp. 32–41.

[103] W. Muła *et al.*, "Faster population counts using avx2 instructions," *The Computer Journal*, 2017.

[104] G. E. Blelloch, "Pre x sums and their applications," Citeseer, Tech. Rep., 1990.

[105] S. Beamer, K. Asanović, and D. Patterson, "The gap benchmark suite," *arXiv preprint arXiv:1508.03619*, 2015.

[106] A. Appleby, "Murmurhash3," https://github.com/aappleby/smhasher, 2016.

[107] R. Chandra, L. Dagum, D. Kohr, R. Menon, D. Maydan, and J. McDonald, *Parallel programming in OpenMP*. Morgan kaufmann, 2001.

[108] M. Rahman and M. A. Hasan, "Sampling triples from restricted networks using mcmc strategy," in *ACM CIKM*, 2014.

[109] T. Hoefler and R. Belli, "Scientific benchmarking of parallel computing systems: twelve ways to tell the masses when reporting performance results," in *Proceedings of the international conference for high performance computing, networking, storage and analysis*, 2015, pp. 1–12.

[110] O. Papapetrou, W. Siberski, and W. Nejdl, "Cardinality estimation and dynamic length adaptation for bloom filters," *Distributed and Parallel Databases*, vol. 28, no. 2, pp. 119–156, 2010.

[111] H. Harmouch and F. Naumann, "Cardinality estimation: An experimental survey," *Proceedings of the VLDB Endowment*, vol. 11, no. 4, pp. 499–512, 2017.

[112] S. Singh and R. Nasre, "Scalable and performant graph processing on gpus using approximate computing," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 3, pp. 190–203, 2018.

[113] Z. Shang and J. X. Yu, "Auto-approximation of graph computing," *VLDB*, vol. 7, no. 14, pp. 1833–1844, 2014.

[114] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.

[115] J. Kunegis, "Konect: the koblenz network collection," in *Proc. of Intl. Conf. on World Wide Web (WWW)*. ACM, 2013, pp. 1343–1350.

[116] C. Demetrescu, A. V. Goldberg, and D. S. Johnson, *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*. American Math. Soc., 2009, vol. 74.

[117] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015. [Online]. Available: http://networkrepository.com

[118] P. Boldi and S. Vigna, "The WebGraph framework I: Compression techniques," in *World Wide Web Conf. (WWW)*, 2004, pp. 595–601.

[119] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 985–1042, 2010.

[120] Z. Bar-Yossef *et al.*, "Counting distinct elements in a data stream," in *RANDOM*, 2002, pp. 1–10.

[121] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier, "Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm," in *Discrete Mathematics and Theoretical Computer Science*, 2007, pp. 137–156.

[122] M. Besta and T. Hoefler, "Survey and taxonomy of lossless graph compression and space-efficient graph representations," *arXiv preprint arXiv:1806.01799*, 2018.

[123] M. Besta, D. Stanojevic, T. Zivic, J. Singh, M. Hoerold, and T. Hoefler, "Log (graph): a near-optimal high-performance graph representation." in *PACT*. ACM, 2018, pp. 7–1. [Online]. Available: https://doi.org/10.1145/3243176.3243198

[124] D. Ediger *et al.*, "Massive streaming data analytics: A case study with clustering coefficients," in *IEEE IPDPSW*, 2010.

[125] S. Galhotra, A. Bagchi, S. Bedathur, M. Ramanath, and V. Jain, "Tracking the conductance of rapidly evolving topic-subgraphs," *Proc. VLDB*, vol. 8, no. 13, pp. 2170–2181, 2015.

[126] M. Besta *et al.*, "Practice of streaming processing of dynamic graphs: Concepts, models, and systems," *IEEE TPDS*, 2022.

[127] A. Saha *et al.*, "Reachability in large graphs using bloom filters," in *IEE ICDEW*, 2019.

[128] T. Eden, D. Ron, and C. Seshadhri, "On approximating the number of k-cliques in sublinear time," *SIAM Journal on Computing*, vol. 49, no. 4, pp. 747–771, 2020.

[129] ——, "Faster sublinear approximation of the number of k-cliques in low-arboricity graphs," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2020, pp. 1467–1478.

[130] A. S. Biswas, T. Eden, and R. Rubinfeld, "Towards a decomposition-optimal algorithm for counting and sampling arbitrary motifs in sublinear time," *arXiv preprint arXiv:2107.06582*, 2021.

[131] T. Washio and H. Motoda, "State of the art of graph-based data mining," *Acm Sigkdd Explorations Newsletter*, vol. 5, no. 1, pp. 59–68, 2003.

[132] V. E. Lee, N. Ruan, R. Jin, and C. Aggarwal, "A survey of algorithms for dense subgraph discovery," in *Managing and Mining Graph Data*. Springer, 2010, pp. 303–336.

[133] S. U. Rehman, A. U. Khan, and S. Fong, "Graph mining: A survey of graph mining techniques," in *Seventh International Conference on Digital Information Management (ICDIM 2012)*. IEEE, 2012, pp. 88–92.

[134] B. Gallagher, "Matching structure and semantics: A survey on graph-based pattern matching." in *AAAI Fall Symposium: Capturing and Using Patterns for Evidence Detection*, 2006, pp. 45–53.

[135] T. Ramraj and R. Prabhakar, "Frequent subgraph mining algorithms-a survey," *Procedia Computer Science*, vol. 47, pp. 197–204, 2015.

[136] C. C. Aggarwal and H. Wang, *Managing and mining graph data*. Springer, 2010, vol. 40.

[137] L. Tang and H. Liu, "Graph mining applications to social network analysis," in *Managing and Mining Graph Data*. Springer, 2010, pp. 487–513.

[138] P. Ribeiro, P. Paredes, M. E. Silva, D. Aparicio, and F. Silva, "A survey on subgraph counting: Concepts, algorithms and applications to network motifs and graphlets," *arXiv preprint arXiv:1910.13011*, 2019.

[139] M. Al Hasan and M. J. Zaki, "A survey of link prediction in social networks," in *Social network data analytics*. Springer, 2011, pp. 243–275.

[140] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[141] F. Cazals and C. Karande, "A note on the problem of reporting maximal cliques," *Theoretical Computer Science*, vol. 407, no. 1-3, pp. 564–568, 2008.

[142] D. Eppstein, M. Löffler, and D. Strash, "Listing all maximal cliques in sparse graphs in near-optimal time," in *Algorithms and Computation - 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I*, 2010, pp. 403–414. [Online]. Available: https://doi.org/10.1007/978-3-642-17517-6_36

[143] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," *Theor. Comput. Sci.*, vol. 363, no. 1, pp. 28–42, 2006. [Online]. Available: https://doi.org/10.1016/j.tcs.2006.06.015

[144] S. Jabbour *et al.*, "Pushing the envelope in overlapping communities detection," in *IDA*. Springer, 2018, pp. 151–163.

[145] Z. Wu *et al.*, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[146] J. Zhou *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[147] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[148] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[149] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy, "Machine learning on graphs: A model and comprehensive taxonomy," *arXiv preprint arXiv:2005.03675*, 2020.

[150] W. L. Hamilton *et al.*, "Representation learning on graphs: Methods and applications," *arXiv preprint arXiv:1709.05584*, 2017.

[151] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[152] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[153] L. Gianinazzi, M. Fries, N. Dryden, T. Ben-Nun, and T. Hoefler, "Learning combinatorial node labeling algorithms," *arXiv preprint arXiv:2106.03594*, 2021.

[154] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *arXiv preprint arXiv:2104.13478*, 2021.

[155] M. Besta and T. Hoefler, "Parallel and distributed graph neural networks: An in-depth concurrency analysis," *arXiv preprint arXiv:2205.09702*, 2022.

[156] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks*, vol. 6, no. 1, pp. 1–23, 2019.

[157] J. de Fine Licht, M. Blott, and T. Hoefler, "Designing scalable fpga architectures using high-level synthesis," *ACM SIGPLAN Notices*, vol. 53, no. 1, pp. 403–404, 2018.

[158] I. Kuon, R. Tessier, and J. Rose, *FPGA architecture: Survey and challenges*. Now Publishers Inc, 2008.

[159] M. Besta, D. Stanojevic, J. D. F. Licht, T. Ben-Nun, and T. Hoefler, "Graph processing on fpgas: Taxonomy, survey, challenges," *arXiv preprint arXiv:1903.06697*, 2019.

[160] J. Cong, H. Huang, C. Ma, B. Xiao, and P. Zhou, "A fully pipelined and dynamically composable architecture of cgra," in *2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines*. IEEE, 2014, pp. 9–16.

[161] O. Mutlu *et al.*, "Processing Data Where It Makes Sense: Enabling In-Memory Computation," *MicPro*, 2019.

[162] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "A modern primer on processing in memory," *arXiv preprint arXiv:2012.03112*, 2020.

[163] S. Ghose, K. Hsieh, A. Boroumand, R. Ausavarungnirun, and O. Mutlu, "The processing-in-memory paradigm: Mechanisms to enable adoption," in *Beyond-CMOS Technologies for Next Generation Computer Design*. Springer, 2019, pp. 133–194.

[164] V. Seshadri, Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu, P. B. Gibbons, and M. A. Kozuch, "Rowclone: fast and energy-efficient in-dram bulk data copy and initialization," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, 2013, pp. 185–197.

[165] J. Gómez-Luna, I. E. Hajj, I. Fernandez, C. Giannoula, G. F. Oliveira, and O. Mutlu, "Benchmarking a new paradigm: An experimental analysis of a real processing-in-memory architecture," *arXiv preprint arXiv:2105.03814*, 2021.

[166] G. F. Oliveira, J. Gómez-Luna, L. Orosa, S. Ghose, N. Vijaykumar, I. Fernandez, M. Sadrosadati, and O. Mutlu, "Damov: A new methodology and benchmark suite for evaluating data movement bottlenecks," *arXiv preprint arXiv:2105.03725*, 2021.

[167] N. Hajinazar, G. F. Oliveira, S. Gregorio, J. D. Ferreira, N. M. Ghiasi, M. Patel, M. Alser, S. Ghose, J. Gómez-Luna, and O. Mutlu, "Simdram: a framework for bit-serial simd processing using dram," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 329–345.

[168] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Ambit: In-memory accelerator for bulk bitwise operations using commodity dram technology," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2017, pp. 273–287.

[169] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A scalable processing-in-memory accelerator for parallel graph processing," in *ISCA*, 2015.

[170] E. de Klerk, D. V. Pasechnik, and J. P. Warners, "On approximate graph colouring and max-k-cut algorithms based on the θ-function," *Journal of Combinatorial Optimization*, vol. 8, no. 3, pp. 267–294, 2004.

[171] S. Khot and O. Regev, "Vertex cover might be hard to approximate to within 2- ε," *Journal of Computer and System Sciences*, vol. 74, no. 3, pp. 335–349, 2008.

[172] J. T. Wang *et al.*, "Algorithms for approximate graph matching," *Information Sciences*, 1995.

[173] L. Gianinazzi, P. Kalvoda, A. De Palma, M. Besta, and T. Hoefler, "Communication-avoiding parallel minimum cuts and connected components," in *ACM SIGPLAN Notices*, vol. 53, no. 1, ACM. ACM New York, NY, USA, 2018, pp. 219–232. [Online]. Available: https://doi.org/10.1145/3200691.3178504

[174] A. Kamath, R. Motwani, K. Palem, and P. Spirakis, "Tail bounds for occupancy and the satisfiability threshold conjecture," *Random Structures & Algorithms*, vol. 7, no. 1, pp. 59–80, 1995.

[175] R. Israel. Variance of the number of empty cells. [Online]. Available: https://math.stackexchange.com/questions/85596/variance-of-the-number-of-empty-cells

[176] M.-T. Chao and W. Strawderman, "Negative moments of positive random variables," *Journal of the American Statistical Association*, vol. 67, no. 338, pp. 429–431, 1972.

[177] J. G. Liao and A. Berg, "Sharpening jensen's inequality," *The American Statistician*, vol. 73, no. 3, pp. 278–281, 2019. [Online]. Available: https://doi.org/10.1080/00031305.2017.1419145

[178] H. A. David and H. N. Nagaraja, *Order statistics*. Wiley, 2003.

[179] K. Beyer *et al.*, "On synopses for distinct-value estimation under multiset operations," ser. SIGMOD, 2007.

Appendix

Here, we provide proofs and various details omitted from the main part of the manuscript.

A. Probability Distributions

Consider a sequence of n trials (experiments). A single trial is a Bernoulli trial, i.e., it gives a *success* or a *failure* outcome with a probability p or $1-p$, respectively. In the context of PG, a single trial will correspond to some property of a given set representation, for example whether a given bit in a BF has 1 or 0. Now, if all the trials are independent (i.e., obtaining a specific outcome does not impact the number of such potential outcomes in future trials), the resulting distribution is **binomial** (commonly denoted as $Bin(n,p)$). Otherwise (i.e., obtaining a specific outcome decreases by one the number of such potential outcomes in future trials), it is **hypergeometric** (usually denoted as $Hyper(N,K,n)$). Both distributions enable deriving specific *probabilities* for the number of either success or failure trials.

B. Plug-In Principle

Some concentration bounds that we present in this paper depend on a given set size (i.e., $|X|$ appears in the formulation of the bound). Thus if we want to obtain an estimate of the upper bound, we need to substitute the estimator $\widehat{|X|}$ instead of $|X|$ whenever $|X|$ appears. This procedure is known as the plug-in principle and it is well established in statistics. However this method is safe to use only if the estimator we substitute is at least consistent for the parameter of interest (i.e. $|X|$). Indeed this is the case for all the BF and MinHash estimators presented in this paper.

C. BF Sketches for Single Sets

We provide extended results for BF for single sets.

1) Concentration Bound for BF Single Sets:

Proposition A.1. *Let $\widehat{|X|}_S$ be the estimator defined in Eq. (1). For $B_X, b \in \mathbb{N}$ such that $b = o(\sqrt{B_X})$, and a set X such that $b|X| \leq 0.499 B_X \log B_X$ the following holds:*

$$E\left[\left(\widehat{|X|}_S - |X|\right)^2\right] \leq (1+o(1))\left(e^{\frac{|X|b}{(B_X-1)}}\frac{B_X}{b^2} - \frac{B_X}{b^2} - \frac{|X|}{b}\right)$$

Proof. We now prove Proposition A.1. Before bounding the mean squared error of $\widehat{|X|}_S$, we need to prove several simple bounds. Let $\mu = E[B_{0,X}] = B_X\left(1-\frac{1}{B_X}\right)^{b|X|}$. It holds:

$$\begin{aligned}\mu &\geq B_X\left(1-\frac{1}{B_X}\right)^{0.499 B_X \log B_X}\\ &\geq B_X \exp\left(-\frac{0.499\log B_X}{1-\frac{1}{B_X}}\right)\\ &= B_X^{0.501-o(1)}\end{aligned}$$

Let us fix some $\varepsilon > 0$. Let $\mathcal{E}$ be the event that $B_{0,X} \geq \mu/(1+\varepsilon)$. [174, Theorem 2] prove that: $P(\mathcal{E}) \geq 1 - \exp(-\Omega(\mu^2/B_X)) \geq 1 - \exp\left(-B_X^{\Omega(1)}\right)$. We have $\widehat{|X|} = -\frac{B_X}{b}\log(B_{X,0}/B_X + \mathbb{I}[B_{X,0} = 0]) \leq B_X \log B_X$ and by our assumption, $|X| \leq b|X| \leq 0.499 B_X \log B_X$. It thus holds $(\widehat{|X|} - |X|)^2 \leq O(B_X^2 \log^2 B_X)$. Let $\kappa = -\frac{B_X}{b}\log\left(1-\frac{1}{B_X}\right)^{b|X|} = -B_X|X|\log\left(1-\frac{1}{B_X}\right)$. Moreover for $B_X \to \infty$, we have $\log(1-1/B_X) = -1/B_X + O(1/B_X^2)$. Therefore, it holds $\kappa = |X| + o(1)$.

Now we are able to bound the mean squared error of $\widehat{|X|}_S$. We present each step of the derivation as a unique figure (see Fig. 10 below) to improve the clarity of the content.

In particular, eq. (10) in Figure 10 holds because for any $a,b,c \in \mathbb{R}$ and $\varepsilon > 0$, it holds[6] $(a-b)^2 \leq (1+\varepsilon)(a-c)^2 + \frac{1+\varepsilon}{\varepsilon}(c-b)^2$. Eq. (13) holds because on $\mathcal{E}$, given $B_{X,0} \geq \mu/(1+\varepsilon)$, $\log(B_{X,0}/B_X)$ is c-lipschitz for $c = (1+\varepsilon)B_X/\mu \leq (1+\varepsilon)e^{\frac{2b|X|}{B_X(1-1/B_X)}} = (1+\varepsilon)e^{\frac{2b|X|}{B_X-1}}$. Eq. (16) holds because

[6] This inequality is equivalent to $(1+\varepsilon)(a-c)^2 + \frac{1+\varepsilon}{\varepsilon}(c-b)^2 - (a-b)^2 \geq 0$. The left-hand side can be simplified to $\frac{(\varepsilon a + b - c(1+\varepsilon))^2}{\varepsilon}$ and the inequality thus holds.

$$
\begin{aligned}
&E[(\widehat{|X|}-|X|)^2] && (8)\\
=&E[(\widehat{|X|}-|X|)^2|\mathcal{E}]P(\mathcal{E})+E[(\widehat{|X|}-|X|)^2|\neg\mathcal{E}]P(\neg\mathcal{E}) && (9)\\
\leq&(1+\varepsilon)E[(\widehat{|X|}-\kappa)^2|\mathcal{E}]+\frac{1+\varepsilon}{\varepsilon}E[(\kappa-|X|)^2|\mathcal{E}]+O(B_X^2\log^2 B_X)\cdot\exp(-B_X^{\Omega(1)}) && (10)\\
\leq&\frac{(1+\varepsilon)B_X^2}{b^2}E[(\log(B_{X,0}/B_X)-\log(1-1/B_X)^{b|X|})^2|\mathcal{E}]+O((\kappa-|X|)^2)+\exp(-B_X^{\Omega(1)}) && (11)\\
\leq&\frac{(1+\varepsilon)B_X^2}{b^2}E[(\log(B_{X,0}/B_X)-\log(1-1/B_X)^{b|X|})^2|\mathcal{E}]+O(|X|/B_X) && (12)\\
\leq&\frac{(1+\varepsilon)^2B_X^2}{b^2}e^{2b|X|/B_X}E[(B_{X,0}/B_X-(1-1/B_X)^{b|X|})^2|\mathcal{E}]+O(|X|/B_X) && (13)\\
\leq&\frac{(1+\varepsilon)^2B_X^2}{b^2}e^{2b|X|/(B_X-1)}\cdot E[(B_{X,0}/B_X-(1-1/B_X)^{b|X|})^2]/P[\mathcal{E}]+O(|X|/B_X) && (14)\\
=&\big((1+\varepsilon)^2+o(1)\big)\frac{B_X^2}{b^2}e^{2b|X|/(B_X-1)}\cdot E[(B_{X,0}/B_X-(1-1/B_X)^{b|X|})^2]+O(|X|/B_X) && (15)\\
=&\big((1+\varepsilon)^2+o(1)\big)\frac{e^{2b|X|/(B_X-1)}}{b^2}Var[B_{X,0}]+O(|X|/B_X) && (16)\\
\leq&\big((1+\varepsilon)^2+o(1)\big)e^{2b|X|/(B_X-1)}\cdot\left(e^{-\frac{b|X|}{B_X}}\frac{B_X}{b^2}-B_X/b^2-|X|/b\right)+O(|X|/B_X) && (17)\\
\leq&\big((1+\varepsilon)^2+o(1)\big)\left(e^{|X|b/(B_X-1)}\frac{B_X}{b^2}-B_X/b^2-|X|/b\right)+O(|X|/B_X) && (18)\\
\leq&\big((1+\varepsilon)^2+o(1)\big)\left(e^{|X|b/(B_X-1)}\frac{B_X}{b^2}-B_X/b^2-|X|/b\right) && (19)
\end{aligned}
$$

Fig. 10: Detailed steps of the derivation of an upper bound for the mean squared error of the BF estimator of the single set size.

$E[B_{X,0}/B_X]=(1-1/B_X)^{b|X|}$ and eq. (17) holds because $Var(B_{X,0})\sim B_Xe^{-\frac{b|X|}{B_X}}-B_X\left(\frac{b|X|}{B_X}+1\right)e^{-\frac{2b|X|}{B_X}}$ [175]. By sending $\varepsilon\to 0$, we get that[7]:

$$
E[(\widehat{|X|}-|X|)^2]\leq(1+o(1))\left(e^{\frac{|X|b}{(B_X-1)}}\frac{B_X}{b^2}-\frac{B_X}{b^2}-\frac{|X|}{b}\right)
$$

□

2) Class of Estimators with General Bounds: The bound on the MSE presented in Proposition A.1 holds up to some assumptions (i.e. $b=o(\sqrt{B_X})$ and $b|X|\leq 0.499B_X\log B_X$) and an $o(1)$ term. To derive a concentration bound for the MSE which does not depend on these conditions and that enhance the interpretability, we develop a *class of estimators* which encompasses the one by Swamidass et al. [59]. To introduce this framework, we first propose a new limiting estimator called $\widehat{|X|}_L$ which belongs to this class. We obtain $\widehat{|X|}_L$ by simplifying the estimator from Eq. (1) and taking the limit for $B_X\to\infty$:

$$
\begin{aligned}
\widehat{|X|}_L\equiv\lim_{B_X\to\infty}\widehat{|X|}_S&=\lim_{B_X\to\infty}-\frac{B_X}{b}\log\left(1-\frac{B_{X,1}}{B_X}\right)\\
&=\log\left(\lim_{B_X\to\infty}\left(1-\frac{B_{X,1}}{B_X}\right)^{-\frac{B_X}{b}}\right)\\
&=\log\left(\exp\left(\frac{B_{X,1}}{b}\right)\right)=\frac{B_{X,1}}{b} \qquad (20)
\end{aligned}
$$

We can perform this simplification thanks to the continuity of the logarithm in $(0,\infty)$ that allows us to safely move the limit inside log, knowing that $B_X,b\in\mathbb{N}$ by construction. This result tells us that, as B_X increases, $\widehat{|X|}_S$ *rescales the number of ones in the BF* by the quantity $\frac{1}{b}$ because $\widehat{|X|}_S\sim\frac{B_{X,1}}{b}$ for X,b fixed and $B_X\to\infty$. We can also prove that $\widehat{|X|}_S\leq\frac{\log B_X}{b}B_{X,1}$ thus implying that $\widehat{|X|}_S$ can *inflate* the number of ones *at most* by the factor $\frac{\log B_X}{b}$. These interesting insights motivate us to propose a general class of estimators. The key idea is to define any BF estimator as a *function of a random variable* (i.e. $B_{X,1}$, the number of ones in a BF). Specifically, we have $\widehat{|X|}_\bullet\equiv\delta_{B_X,b}(B_{X,1})$, where $\delta(\cdot)$ is a given non-negative function of B_X, b, and $B_{X,1}$. We choose to denote $\delta_{B_X,b}(B_{X,1})$ and $\delta_{B_X,b}$ instead of the usual $\delta(B_X,b,B_{X,1})$ and $\delta(B_X,b)$ to clearly separate the deterministic BF design parameters B_X and b from the unique random component $B_{X,1}$. The key benefit of this formulation is that (1) it generalizes both $\widehat{|X|}_S$ and $\widehat{|X|}_L$, and (2) we can

[7] It is well known that if $f(x)\leq(1+\varepsilon)g(x)$ for any $\varepsilon>0$, then $f(x)\leq(1+o(1))g(x)$.

use it to provide concentration bounds that are applicable to $\widehat{|X|}_L$, and many other estimators within the proposed class depending on the functional form of $\delta_{B_X,b}(B_{X,1})$. To obtain $\widehat{|X|}_S$, we set:

$$\widehat{|X|}_S \equiv \delta_{B_X,b}(B_{X,1}) = \frac{B_X}{b}\log\left(1-\frac{B_{X,1}}{B_X}\right).$$

To recover $\widehat{|X|}_L$, we first (with a slight abuse of notation) fix $\delta_{B_X,b}(B_{X,1})$ to be linear in $B_{X,1}$ and then set it to be specifically equal to $\frac{1}{b}$:

$$\widehat{|X|}_L \equiv \delta_{B_X,b} \cdot B_{X,1} = \frac{B_{X,1}}{b} \tag{21}$$

We underline that if $\delta_{B_X,b}(B_{X,1})$ is linear in $B_{X,1}$ we are implicitly imposing, depending on the values of B_X and b, either a *deflation* or an *inflation* of the observed *number of ones* in the BF. For example, we have already seen that, when $B_X \to \infty$ for fixed X, b, we have $\delta_{B_X,b}(B_X) = \frac{B_X}{b}$ in Eq. (1). We now show that any estimator that can be written as $\widehat{|X|}_\bullet$ with $\delta_{B_X,b}(B_{X,1})$ linear in $B_{X,1}$ has a bounded MSE.

Proposition A.2. *Let* $\widehat{|X|}_\bullet \equiv \delta_{B_X,b} \cdot B_{X,1}$. *For* $B_X, b \in \mathbb{N}$, *the following holds:*

$$E\left[\left(\widehat{|X|}_\bullet - |X|\right)^2\right] \leq \left[|X| - \ \delta_{B_X,b}\, B_X\left(1 - e^{-\frac{|X|b}{B_X}}\right)\right]^2$$
$$+\ \delta^2_{B_X,b}\, B_X\left[e^{-\frac{|X|b}{B_X}} - \left(1+\frac{|X|b}{B_X}\right)e^{-\frac{2|X|b}{B_X}}\right]$$

We use Chebyshev's inequality to get the final concentration bound:

$$P\left(\left|\widehat{|X|}_\bullet - |X|\right| \geq t\right) \leq \frac{\left[|X| - \ \delta_{B_X,b}\, B_X\left(1 - e^{-\frac{|X|b}{B_X}}\right)\right]^2}{t^2}$$
$$+ \frac{\delta^2_{B_X,b}\, B_X\left[e^{-\frac{|X|b}{B_X}} - \left(1+\frac{|X|b}{B_X}\right)e^{-\frac{2|X|b}{B_X}}\right]}{t^2}$$

By fixing $\delta_{B_X,b} = \frac{1}{b}$, we obtain a valid bound for $\widehat{|X|}_L$ which is the limiting estimator we present in our evaluation (Section VIII).

Proof. We provide a proof of proposition A.2. We start by the well known MSE decomposition:

$$E\left[\left(\widehat{|X|}_\bullet - |X|\right)^2\right] = E\left[\left(\widehat{|X|}_\bullet - |X|\right)\right]^2 + \ Var(\widehat{|X|}_\bullet) \tag{22}$$

Now notice that $E[B_{0,X}] = B_X\left(1-\frac{1}{B_X}\right)^{b|X|}$. Then, since $\widehat{|X|}_\bullet = \delta_{B_X,b}\ B_{X,1}$, we can easily derive:

$$E\left[\delta_{B_X,b}\ B_{X,1}\right] = E\left[\delta_{B_X,b}\ (B_X - B_{X,0})\right]$$
$$= \delta_{B_X,b}\ B_X\left[1 - \left(1-\frac{1}{B_X}\right)^{b|X|}\right]$$

On the other hand, to bound the variance of the simplified estimator, we follow the same reasoning outlined in Proposition A.1. Indeed it holds that $Var(B_{X,0}) \sim B_X\left[e^{-\frac{|X|b}{B_X}} - \left(1+\frac{|X|b}{B_X}\right)e^{-\frac{2|X|b}{B_X}}\right]$ as shown in [175]. Now notice that $Var(B_{X,1}) = Var(B_X - B_{X,0}) = Var(B_{X,0})$. At this point we can substitute in eq. (22) the squared bias and variance of $\widehat{|X|}_\bullet$ to conclude that:

$$E\left[\left(\widehat{|X|}_\bullet - |X|\right)^2\right] \leq \left\{|X| - \delta_{B_X,b}\, B_X\left[1 - \left(1-\frac{1}{B_X}\right)^{b|X|}\right]\right\}^2$$
$$+\delta^2_{B_X,b}\, B_X\left[e^{-\frac{|X|b}{B_X}} - \left(1+\frac{|X|b}{B_X}\right)e^{-\frac{2|X|b}{B_X}}\right]$$

which ends the proof. To improve the interpretability of the bound, we use the fact that $\left(1-\frac{1}{B_X}\right)^{b|X|} \sim e^{-\frac{|X|b}{B_X}}$ in the statement of Proposition A.2.

□

3) Enhancing the Estimator by Swamidass [59]: The estimator by Swamidass et al. [59], is *divergent*[8] in its original form. To alleviate this, we replace $B_{X,1}$ with $\widetilde{B}_{X,1} \equiv B_{X,1} - \mathbb{I}[B_{X,1} = B_X]$, where, for a given proposition P, $\mathbb{I}[P]$ is 1 if P holds, and 0 otherwise. $\widetilde{B}_{X,1}$ only differs from $B_{X,1}$ in the unlikely case of $B_{X,1} = B_X$. Thanks to this modification, our estimator $\widehat{|X|}$ has, unlike Swamidass et al.'s, a finite expectation (as it is bounded).

The final form of the estimator is

$$\widehat{\widetilde{|X|}} = -\frac{B_X}{b}\log\left(1-\frac{\widetilde{B_{X,1}}}{B_X}\right)$$

4) Proof of consistency and asymptotic unbiasedness: We need to show that $\widehat{|X|}_S = -\frac{B_X}{b}\log\left(1-\frac{B_{X,1}}{B_X}\right)$ is consistent and asymptotically unbiased as $B_X \to \infty$. We provide here an intuitive formulation based on the false positive probability which can be easily made more rigorous by direct application of the definition of consistency which we omit for the sake of simplicity. First of all, as shown in eq.(21), we can notice that $\widehat{|X|}_S \sim \widehat{|X|}_L$ as the Bloom Filter size diverges. This means that the proof is valid for both estimators because they are asymptotically equivalent. Now we can look at the probability of false positives as $B_X \to \infty$ for fixed and finite b and $|X|$:

$$\lim_{B_X\to\infty}\left[1-\left(1-\frac{1}{B_X}\right)^{b|X|}\right]^b \to 0$$

The result above tells us that false positive matches cannot happen anymore in the limit. Each element of $|X|$ will then be hashed in a *personal* bit and counting the number of ones in B_X (and dividing by b in case of multiple hash functions) will always deliver $|X|$ at a given precision as $|X|$ is fixed and $B_X \to \infty$. Thus we can conclude that $\frac{B_{X,1}}{b} \xrightarrow{p} |X|$ which proves consistency. Asymptotic unbiasedness follows from

[8] An estimator whose moments are not finite. In the case of the estimator Swamidass et al. [59], the expectation of $\widehat{|X|}$, and thus also the higher moments, diverge, which happens for $B_{X,1} = B_X$

consistency in our case as the variance of both estimators is bounded (see the proof of Proposition IV.1). The same reasoning can be easily extended to show consistency and asymptotic unbiasedness also for $|\widehat{X \cap Y}|_{AND}$ and $|\widehat{X \cap Y}|_{OR}$ presented in section IV-B.

D. Proposition IV.1

Proof. To prove Proposition IV.1 from Section IV-B we can extend in a straightforward way the proof presented for Proposition A.1. Indeed we just need to substitute $|X|$ with $|X \cap Y|$ and B_X with $B_{X \cap Y}$ to obtain the desired result. □

E. MinHash Sketches for Set Intersection

1) Expectation formula: Since in the case of k-hash, $|M_X \cap M_Y| \sim Bin(\ k\ ,\ J_{X,Y}\ )$, and for 1-hash, $|M^1_X \cap M^1_Y| \sim Hypergeometric(|X \cup Y|, |X \cap Y|, k)$, we have:

$$\mathbb{E}[|\widehat{X \cap Y}|_{kH}] = (|X| + |Y|) \sum_{s=0}^{k} \binom{k}{s} (J_{X,Y})^s (1 - J_{X,Y})^{k-s} \frac{s}{k+s} \tag{23}$$

$$\mathbb{E}[|\widehat{X \cap Y}|_{1H}] = (|X| + |Y|) \sum_{s=0}^{k} \frac{\binom{|X \cap Y|}{s}\binom{|X \cup Y| - |X \cap Y|}{k-s}}{\binom{|X \cup Y|}{k}} \frac{s}{k+s} \tag{24}$$

There exists an involved closed form expression for equation (23) which is beyond the scope of this paper. We refer the interested reader to [176] for a clear derivation of a similar problem.

2) Proof of consistency and asymptotic unbiasedness: We start to show that $|\widehat{X \cap Y}|_{kH}$ is consistent. This follows respectively from Proposition IV.2 statement. Indeed by taking the limit for $k \to \infty$ with fixed and finite $|X|$ and $|Y|$ we obtain:

$$\lim_{k \to \infty} P\left(\left| |\widehat{X \cap Y}|_{kH} - |X \cap Y| \right| \geq t \right) \leq \lim_{k \to \infty} 2e^{-\frac{2\ k\ t^2}{(|X|+|Y|)^2}} \to 0$$

The above implies that $|\widehat{X \cap Y}|_{kH} \xrightarrow{p} |X \cap Y|$. On the other hand, for $|\widehat{X \cap Y}|_{1H}$ we are in the *sampling without replacement* scheme. This means that the population size (i.e. $|X \cup Y|$) is finite and by taking the limit for $k \to |X \cup Y|$ in Proposition IV.3, with fixed and finite $|X|$ and $|Y|$, we have already sampled the entire population contrarily to the k-Hash case. Thus $|\widehat{X \cap Y}|_{1H}$ is also a consistent estimator of $|X \cap Y|$. Then, for both estimators, the asymptotic unbiasedness follows from consistency and by noticing that both $|\widehat{X \cap Y}|_{kH}$ and $|\widehat{X \cap Y}|_{1H}$ have a bounded variance.

3) Sub-Gaussian preliminaries: We recall some key notions of sub-gaussian random variables as they are necessary for the following proofs. First of all, we define $\psi_X(\lambda) = \log(\mathbb{E}[e^{\lambda X}])$ as the logarithmic moment generating function (i.e. cumulant) of a generic random variable X. For example, if Z is a centered normal random variable with variance σ^2, we have that $\psi_Z(\lambda) = \frac{\lambda^2 \sigma^2}{2}$. It can be shown, we refer the interested reader to chapter 2 of [38] for a detailed explanation, that Chernoff's inequality in this case implies, for all $t > 0$, that:

$$P(Z \geq t) \leq e^{-\frac{t^2}{2\sigma^2}} \tag{25}$$

The bound above, characterize the decay of the tail probabilities of a centered normal random variable. If the tail probabilities of a generic centered random variable X, decrease at least as rapidly as the ones in (25) then X is *sub-gaussian*. More formally, a centered random variable X is said to be *sub-gaussian* with variance factor σ^2 if:

$$\psi_X(\lambda) \leq \frac{\lambda^2 \sigma^2}{2} \quad \forall \lambda \in \mathbb{R} \tag{26}$$

We underline that (26) only requires $Var(X) \leq \sigma^2$. Moreover, if we call $\mathcal{G}(\sigma^2)$ the collection of random variables for which (26) holds (e.g. all bounded random variables belongs to $\mathcal{G}(\sigma^2)$), we can state that:

Lemma A.3. *Let $X_1, \ldots, X_n$ be sub-gaussians random variables so that $X_i \in \mathcal{G}(\sigma_i^2)$ for every $i \in \{1, \ldots, n\}$. Then $\sum_{i=1}^n X_i \in \mathcal{G}((\sum_{i=1}^n \sigma_i)^2)$. Moreover if the random variables are independent, then $\sum_{i=1}^n X_i \in \mathcal{G}(\sum_{i=1}^n \sigma_i^2)$.*

This is due to the fact that (26) implies a bound on the moment generating function whose properties, together with the Hölder inequality, help to verify the above statement. For a detailed proof of lemma A.3 see theorem 2.7 in [**?**] while for alternative characterizations of sub-gaussianity in terms of growth of moments, we refer to chapter 2 of [38].

4) Concentration bounds for k-Hash and 1-Hash: We present below the proof of Propositions IV.2 and IV.3. First, we show the following lemma which we will also use later.

Lemma A.4.

$$P(|\hat{J}_1 - J| \geq t), P(|\hat{J}_k - J| \geq t) \leq 2e^{-2t^2 k} \tag{27}$$

Proof. The random variables $k\hat{J}_1$ and $k\hat{J}_k$ follow the hypergeometric and binomial distributions, respectively. Applying the Hoeffding's inequalities in the binomial case, we get the desired inequality. The Serfling's bound can be applied in the case of the hypergeometric distribution. The Serfling's bound always gives better bounds than the Hoeffding's, proving the inequality for $\hat{J}_1$. □

We now show concentration of the sum of the estimators and, therefore also of the individual estimators (by fixing $n = 1$).

Theorem A.5. *Let $Y_1 = \sum_i^n C_i \frac{\hat{J}_1}{1+\hat{J}_1}, Y_k = \sum_i^n C_i \frac{\hat{J}_k}{1+\hat{J}_k}$. Then for any non-negative constants C_i and $S = \sum_{i=1}^n C_i \frac{J}{1+J}$*

$$P(|Y_1 - S| > t), P(|Y_k - S| > t) \leq 2 \exp\left(-\frac{2\ k\ t^2}{(\sum_i^n C_i)^2} \right) \tag{28}$$

Proof. The function $\frac{X}{1+X}$ is 1-Lipschitz and it, therefore, holds that $\forall\ X, X' \in [0, 1]$

$$\left| \frac{X}{1+X} - \frac{X'}{1+X'} \right| \leq |X - X'|$$

The concentration result from Lemma A.4 then also holds for $\frac{\hat{J}_1}{1+\hat{J}_1}$ and $\frac{\hat{J}_k}{1+\hat{J}_k}$. The random variables

$$\frac{\hat{J}_1}{1+\hat{J}_1} - \frac{J}{1+J}$$
$$\frac{\hat{J}_k}{1+\hat{J}_k} - \frac{J}{1+J}$$

are therefore sub-gaussian with variance factor $\sigma^2 = \frac{1}{4k}$ (see eq. (25), (26) and in general section E3). Now we multiply by C_i each variable, we take the sum and, thanks to lemma A.3, we get that $Y_1 - S$ and $Y_k - S$ are sub-gaussian with variance factor

$$\frac{(\sum_{i=1}^{n} C_i)^2}{4k}$$

The theorem follows from the definition of a sub-gaussian random variable (see section E3 for further references). □

We stress here that $\widehat{|X \cap Y|}_{kH}$ derived with k-hash can also be interpreted as a *maximum likelihood estimator (MLE)* (cf. § II-E) for $|X \cap Y|$ because of the invariance property outlined in § II-F and detailed in Chapter 7 of [39]. Indeed since $|M_X \cap M_Y| \sim Bin(k, J_{X,Y})$ we have that $\widehat{J_{X,Y}}_{kH} = \frac{|M_X \cap M_Y|}{k}$ is the maximum likelihood estimator of $J_{X,Y}$ if we assume that the k hash functions are independent and perfectly random (a usual assumption). Then our estimator $\widehat{|X \cap Y|}_{kH}$ is just a function of $\widehat{J_{X,Y}}_{kH}$ and, because of the invariance of the MLE (see *Theorem 7.2.10* in [39]), this implies that $\widehat{|X \cap Y|}_{kH}$ inherits all the properties of this class of estimators. In particular, it is consistent and asymptotically efficient since it reaches the *Cramér-Rao Lower Bound* (see *Theorem 7.3.9* in [39]) meaning that no other estimator can have a lower variance. It is also normally distributed, as k increase, which is useful in general to derive confidence intervals.

F. Results & Derivations for Triangle Counts

1) Proof of consistency and asymptotic unbiasedness: Any estimator for triangle count analyzed in PG, is simply a sum of cardinalities $\widehat{|X \cap Y|}$ for different neighborhoods X and Y (cf. Section III):

$$\widehat{TC}_\star = \frac{1}{3} \sum_{(u,v) \in E} \widehat{|N_u \cap N_v|}_\star$$

where $\star$ indicates a specific $\widehat{|X \cap Y|}_\star$ estimator (cf. Table II). Since we have already proven consistency and asymptotic unbiasedness for each of the $\widehat{|X \cap Y|}_\star$ estimators presented in PG, we now can address jointly the consistency of the triangle count estimators. To do so we just need to acknowledge the fact that a sum of consistent estimators is itself a consistent estimator. Indeed this is a direct consequence of the more general *Slutsky theorem* which enable us to state that $\widehat{TC}_\star \xrightarrow{p} TC$. The asymptotic unbiasedness then follows from consistency and by noticing that all $\widehat{TC}_\star$ estimators have a bounded variance (see all the proofs presented below).

2) Bloom Filters: We first present the estimator $\widehat{|X \cap Y|}_{OR}$. This estimator was introduced before [59] and uses the single set estimator evaluated on the set union:

$$\widehat{|X \cap Y|}_{OR} = |X| + |Y| + \frac{B_{X \cup Y}}{b} \log\left(1 - \frac{B_{X \cup Y,1}}{B_{X \cup Y}}\right) \quad (29)$$

Note that, to obtain the expression above, we also use the fact that $|X \cup Y| = |X| + |Y| - |X \cap Y|$.
We now prove the triangle count bound for BF stated in theorem VII.1 for the triangle count OR estimator (the proof is of course valid also for $\widehat{TC}_{AND}$).

Proof. We first define the mean squared error (mse) as follows:

$$E[(TC - \widehat{TC}_{OR})^2] = (E[\widehat{TC}_{OR}] - TC)^2 + Var(\widehat{TC}_{OR})$$

where the equality is a standard identity.

Now we bound the first component of the mse which is the squared bias of our estimator. In order to ease the notation from now on we denote $|N_u \cup N_v| \ \forall \ (u,v) \in E$ as $|X|_i \ \forall \ i = 1,..,m$ where m is the number of edges. In the same fashion, we denote $\widehat{|X|}_i \ \forall \ i = 1,..,m$ as the estimator of $|N_u \cup N_v| \ \forall \ (u,v) \in E$. Thus we can write:

$$(E[\widehat{TC}_{OR}] - TC)^2$$
$$= \frac{1}{9}\left[\sum_{i=1}^{m} E(\widehat{|X|}_i) - |X|_i\right]^2 \quad (30)$$
$$\leq \frac{1}{9}\left\{\sum_{i=1}^{m}\sum_{j=1}^{m}\left|[E(\widehat{|X|}_i) - |X|_i][E(\widehat{|X|}_j) - |X|_j]\right|\right\} \quad (31)$$
$$\leq \frac{1}{9}\left\{\sum_{i=1}^{m}\sum_{j=1}^{m}\left|[E(\widehat{|X|}_i) - |X|_i]\right|\left|[E(\widehat{|X|}_j) - |X|_j]\right|\right\} \quad (32)$$
$$= \frac{1}{9}\left\{\sum_{i=1}^{m}\sum_{j=1}^{m}\sqrt{[E(\widehat{|X|}_i) - |X|_i]^2}\sqrt{[E(\widehat{|X|}_j) - |X|_j]^2}\right\} \quad (33)$$
$$\leq \frac{m^2}{9}\,(1 + o(1))\left(e^{2\Delta b/(B_X - 1)}\frac{B_X}{b^2} - \frac{B_X}{b^2} - \frac{2\Delta}{b}\right) \quad (34)$$

where (32) follows by Cauchy–Schwarz inequality and (34) by Proposition A.1 which in general bounds the mse (and thus the squared bias by Jensen's inequality) if $2b\Delta \leq 0.499 B_X \log B_X$ where $B_X = min(B_{N_u \cup N_v})$ with $(u,v) \in E$. In particular we underline that any bound obtained by Proposition A.1, which is valid for a given set size, is also automatically valid for all the set sizes smaller than that *a fortiori*. Thus we can notice that $2\Delta \geq |X|_i \ \forall \ i = 1,..,m$ where Δ is the maximum degree of the input graph which justifies (34). At this point, it remains to bound the second component of the mse which is the variance of our estimator. Indeed we can write:

$$
\begin{aligned}
&Var(\widehat{TC}_{OR})\\
&= \frac{1}{9}\, Var\left[\sum_{i=1}^{m} \widehat{|X|}_i\right] && (35)\\
&= \frac{1}{9}\, \sum_{i=1}^{m}\sum_{j=1}^{m} Cov(\widehat{|X|}_i, \widehat{|X|}_j) && (36)\\
&\leq \frac{1}{9}\, \sum_{i=1}^{m}\sum_{j=1}^{m} \sqrt{Var(\widehat{|X|}_i)}\sqrt{Var(\widehat{|X|}_j)} && (37)\\
&\leq \frac{m^2}{9}\,(1+o(1))\left(e^{2\Delta b/(B_X-1)}\frac{B_X}{b^2}-\frac{B_X}{b^2}-\frac{2\Delta}{b}\right) && (38)
\end{aligned}
$$

where (37) holds because of the covariance inequality and (38) by Proposition A.1 which in general bounds the mse (and thus the variance *a fortiori*) if $2b\Delta \leq 0.499 B_X \log B_X$. The justification of (38) is similar to the one outlined before for the squared bias case.

Now, again assuming $2b\Delta \leq 0.499 B_X \log B_X$, we can obtain the bound for the mean squared error of the OR estimator:

$$
\begin{aligned}
&E[(TC-\widehat{TC}_{OR})^2] = \\
&(E[\widehat{TC}_{OR}]-TC)^2 + Var(\widehat{TC}_{OR})\\
&\leq \frac{2m^2}{9}\,(1+o(1))\left(e^{2\Delta b/(B_X-1)}\frac{B_X}{b^2}-\frac{B_X}{b^2}-\frac{2\Delta}{b}\right)
\end{aligned}
$$

The above bound is valid also for $\widehat{TC}_{AND}$ since $\Delta \geq |N_u \cap N_v|\ \forall\ (u,v) \in E$ however it can be made tighter for the same reason. Indeed if $b\Delta \leq 0.499 B_X \log B_X$ (where now $B_X = min(B_{N_u \cap N_v})$ with $(u,v) \in E$) by Chebychev inequality:

$$
\begin{aligned}
&P\left(\left|TC-\widehat{TC}_{AND}\right| \geq t\right) \leq \\
&\frac{2m^2(1+o(1))\left(e^{\Delta b/(B_X-1)}\frac{B_X}{b^2}-\frac{B_X}{b^2}-\frac{\Delta}{b}\right)}{9\, t^2}
\end{aligned}
$$

which is the statement of Theorem VII.1 for the BF case. □

3) MinHash: We can show the concentration of the sum of the set intersection estimators using theorem A.5 presented in Appendix E. Then for the edge $e_i = uv$, we define $C_i = \deg(u)+\deg(v)$ thus giving us $S = \frac{1}{3}\ \sum_{i=1}^{n} C_i \frac{J}{1+J} = TC$. We will not consider the scaling factor $\frac{1}{3}$ till the final expressions of the bounds to ease the notation. Thus we can write:

$$
\begin{aligned}
\sum_{i=1}^{m} C_i = \sum_{i=1, e_i=uv}^{m} \deg(u)+\deg(v) =\\
= \sum_{v\in V} \deg(v)^2
\end{aligned}
$$

Combining the above result with theorem A.5, we obtain the triangle count bound for MinHash presented in Theorem VII.1.

However this bound can be improved if we assume more independence, which will be satisfied in the case of triangle counting when the maximum degree is not too large. We now prove a tighter bound under these conditions.

Theorem A.6. *Let* $Y_1 = \sum_i^n C_i \frac{\hat{J}_1}{1+\hat{J}_1}, Y_k = \sum_i^n C_i \frac{\hat{J}_k}{1+\hat{J}_k}$, *and assume we partition the set of estimators into groups* $\mathcal{X}_1, \cdots, \mathcal{X}_\chi$ *such that estimators from each set are mutually independent. Then for any non-negative constants* C_i *and* $S = \sum_{i=1}^n C_i \frac{J}{1+J}$

$$
\begin{aligned}
P(|Y_1 - S| > t), P(|Y_k - S| > t) &\leq 2\exp\left(-\frac{k(\max(0, t-2S/k))^2}{2(\sum_i^\chi \sqrt{\sum_{d\in\mathcal{X}_i} C_d^2})^2}\right)\\
&\leq 2\exp\left(-\frac{k(\max(0, t-2S/k))^2}{2\chi\sum_i^n C_i^2}\right)
\end{aligned}
$$

Proof. We modify the proof of Theorem A.5 by instead considering the random variables

$$
\begin{aligned}
&\frac{\hat{J}_1}{1+\hat{J}_1} - \frac{J}{1+J} - \mu_1\\
&\frac{\hat{J}_k}{1+\hat{J}_k} - \frac{J}{1+J} - \mu_k
\end{aligned}
$$

where μ_1 and μ_k are chosen so as to make this random variable have mean zero.

We then first sum estimators from each group separately using Lemma A.3, which gives us subgaussian coefficient (i.e. the square root of the variance factor) of $\sqrt{\sum_{d\in\mathcal{X}_i} C_d^2}$. Adding the groups together, we get using again Lemma A.3, that the subgaussian coefficient is $\sigma_{\mathcal{X}} = \sum_i^\chi \sqrt{\sum_{d\in\mathcal{X}_i} C_d^2}$. To finish the proof of the first inequality, we have to show a bound on $\sum C_i\mu_1$ and $\sum C_i\mu_k$. We show the argument for the case of 1-hash, the argument for k-hash is analogous. Note that μ_1 is the jensen gap of $\frac{\hat{J}_1}{1+\hat{J}_1}$. Since Var$(\hat{J}_1) \leq J/k$, by Theorem 1 from [177], we have $-J/k \leq E[\frac{\hat{J}_k}{1+\hat{J}_k}] - \frac{J}{1+J} \leq 0$. We can bound $-J/k \geq 2/k\frac{J}{1+J}$. Therefore, we can bound

$$
-2S/k \leq \sum C_i\mu_1 \leq 0
$$

To prove the second inequality, we define the following optimization problem

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{n} \sqrt{x_n}\\
\text{subject to} \quad & \sum x_i = c
\end{aligned}
$$

Set $x_i = \sum_{d\in\mathcal{X}_i} C_d^2$ and $c = \sum C_i^2$. We see that for every possible assignment of the estimators to the sets $\{\mathcal{X}_i\}_{i=1}^\chi$, we have a feasible solution with objective value equal to the subgaussian coefficient $\sigma_{\mathcal{X}}$. Therefore, the subgaussian coefficient for any assignment to the groups is dominated by the optimum of this optimization problem.

Optimum of this optimization problem is when all x_i's have the same value – otherwise one can pick i, j such that $x_i < x_j$ and $0 < \varepsilon \leq (x_j - x_i)/2$ and then replace x_i by $x_i + \varepsilon$ and similarly x_j by $x_j - \varepsilon$, increasing the objective while retaining feasibility. This gives us objective value of $\chi\sqrt{\sum_{i=1}^{n} C_i^2/\chi} = \sqrt{\chi \sum_i^n C_i^2}$ □

To show the final expression of the bound, we use Theorem A.6. Then by Vizing's theorem, $\chi \le \Delta + 1$ and by the same substitution done for the first bound, we have:

$$\sum_{i=1}^{m} C_i^2 = \sum_{i=1, e_i = uv}^{m} (\deg(u) + \deg(v))^2 \le$$
$$\le \sum_{i=1, e_i = uv}^{m} 2(\deg(u)^2 + \deg(v)^2) = 2\sum_{v \in V} \deg(v)^3$$

Indeed combining the above result with Theorem A.6, we obtain the triangle count bound for MinHash presented in Theorem VII.1 if the maximum degree is Δ.

G. KMV Sketches

1) Single Sets: We state an existing result on the KMV sketching; we use it later to provide a KMV sketch for $|X \cap Y|$ [178]. The hash function used with a KMV maps elements from X to real numbers in $(0, 1]$ u.a.r.[9]. Thus, the hashes should be evenly spaced and one can estimate $|X|$ by dividing the size $k - 1$ of K_X by the largest hash in K_X.

$$\widehat{|X|}_K = \frac{k-1}{\max\{x | x \in K_X\}} \tag{39}$$

As noted in [178, §2.1], the k-th smallest value follows the beta distribution $Beta(\alpha, \beta)$ with shape parameters $\alpha = k$ and $\beta = |X| - k + 1$. Now we can get concentration bounds for the estimator: indeed, following [179], we can show that:

Proposition A.7. *Consider $\widehat{|X|}_K$ in Eq. (39), then the probability of deviation from the true set size, at a given distance $t \ge 0$, is*

$$P\left(\left|\widehat{|X|} - |X|\right| \le t\right) = I_{u(|X|,k,t/|X|)}(k, |X| - k + 1) - I_{l(|X|,k,t/|X|)}(k, |X| - k + 1)$$

where $u(|X|, k, t/|X|) = \frac{k-1}{|X|-t}$ and $l(|X|, k, t/|X|) = \frac{k-1}{|X|+t}$ and $I_x(a, b)$ is the regularized incomplete beta function.

In the case of a KMV estimator bound, we can evaluate:

$$I_x(k, |X| - k + 1) = \sum_{i=k}^{|X|} \binom{|X|}{i} x^i (1 - x)^{|X| - i}$$

2) Set Intersection $|X \cap Y|$: Given $\mathcal{K}_X$ and $\mathcal{K}_Y$ of size k_X and k_Y, one can construct a KMV $\mathcal{K}_{X \cup Y}$ by taking the $k = \min\{k_X, k_Y\}$ smallest elements from $K_X \cup K_Y$. $\widehat{|X \cup Y|}_K$, $\widehat{|X|}_K$ and $\widehat{|Y|}_K$ can be computed using the following equations (note that the second one uses the exact sizes of X, Y instead of their estimators).

$$\widehat{|X \cap Y|}_K = \widehat{|X|}_K + \widehat{|Y|}_K - \widehat{|X \cup Y|}_K \tag{40}$$

$$\widehat{|X \cap Y|}_K = |X| + |Y| - \widehat{|X \cup Y|}_K \tag{41}$$

We present now a simple upper bound (using the union bound) on the probability that $\widehat{|X \cap Y|}_K$ deviates by more than t from the true value.

Proposition A.8. *Let $\widehat{|X \cap Y|}_K$ be the estimator defined in (40), then the following upper bound for the probability of deviation from the true intersection set size, at a given distance $t \ge 0$, holds:*

$$P\left(||\widehat{X \cap Y}|_K - |X \cap Y|| \ge t\right) \le P(||\widehat{X}|_K - |X|| \ge t/3) +$$
$$P(||\widehat{Y}|_K - |Y|| \ge t/3) + P(||\widehat{X \cup Y}|_K - |X \cup Y|| \ge t/3)$$

where the probabilities on the right can be evaluated with Proposition A.7.

Yet, if we know the exact size of X and Y (a reasonable assumption for graph algorithms as the degrees can be easily precomputed), we can get a considerably better bound.

Proposition A.9. *Let $\widehat{|X \cap Y|}_K$ be the estimator from (41), then*

$$P\left(||\widehat{X \cap Y}|_K - |X \cap Y|| \ge t\right) =$$
$$I_{u(|X \cup Y|, k, t/|X \cup Y|)}(k, |X \cup Y| - k + 1)$$
$$- I_{l(|X \cup Y|, k, t/|X \cup Y|)}(k, |X \cup Y| - k + 1)$$

The bound presented above is a simple application of the identity $|X \cap Y| = |X| + |Y| - |X \cup Y|$ and Proposition A.7 on the estimator of $|X \cup Y|$.

[9] uniformly at random